AD-A054 694 ARINC RESEARCH CORP SANTA ANA CALIF WESTERN DIV F/G 9/5

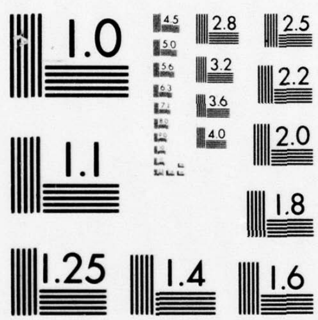
PROGRAMMER'S MANUAL FOR SNAP II COMPUTER PROGRAM.(U)

AUG 68 T D PRICE, C KIMME

UNCLASSIFIED 474-01-2-916 NL

1 OF 2

AD A054694


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

4

PROGRAMMER'S MANUAL FOR SNAP II
COMPUTER PROGRAM

August 1968

Prepared for

U. S. NAVAL WEAPONS CENTER
China Lake, California

Publication 474-01-2-916

ARINC RESEARCH CORPORATION

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 474-01-2-916	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PROGRAMMER'S MANUAL FOR SNAP II COMPUTER PROGRAM		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER 474-01-2-916
7. AUTHOR(s) T.D. Price Carolyn Kimme		8. CONTRACT OR GRANT NUMBER(s) Not Listed
9. PERFORMING ORGANIZATION NAME AND ADDRESS ARINC Research Corporation 2551 Riva Road Annapolis, Maryland 21401		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. NAVAL WEAPONS CENTER China Lake, California		12. REPORT DATE August 1968 13. NUMBER OF PAGES 130
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) U.S. NAVAL WEAPONS CENTER China Lake, California		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) UNCLASSIFIED/UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

4

6

PROGRAMMER'S MANUAL FOR SNAP II COMPUTER PROGRAM

11 Aug[redacted]68

12 175 p.

Prepared for

U. S. NAVAL WEAPONS CENTER
China Lake, California

Prepared by

T D Price Carolyn Kimme

10 T. D. Price Carolyn Kimme

Approved by

J R Gliessman

J. R. Gliessman

ARINC RESEARCH CORPORATION
Western Division
P. O. Box 1375
Santa Ana, California 92702

14 Publication 474-01-2-916

SUMMARY

SNAP II performs 5 main functions in solving a circuit.

This manual provides a detailed technical description of the SNAP II computer program. The manual is designed such that a qualified programmer cannot only gain a full working knowledge of the programming logic, but also alter or add to it by following specific guidelines.

Included in the manual are: ~~1)~~ a description of the main program, subprograms, interrelations between programs, and each of the variables (code words) used in the programs; ~~2)~~ a cross-reference indicating the program in which each variable is used; and ~~3)~~ flow charts depicting each logical sequence in the overall program.

See also the User's manual, AD-A054 696.

100 °C

TABLE OF CONTENTS

1. MAIN PROGRAM AND INTERRELATIONSHIP WITH SUBROUTINES

SNAP II performs five main functions in solving a circuit, as discussed individually in Sections 1.1 through 1.5.

1.1 READING IN CIRCUIT ELEMENT VALUES

Circuit-element values are read into the main program after data statements have been entered and dimensioning has been completed. The values are read in one at a time, and each is checked for logical and keypunching errors by subroutine CHECK.

The node with the greatest numerical value in the circuit is stored as circuit elements are added. An incorrect circuit element value will not stop the program, although errors in card format will activate a diagnostic routine. Circuit Element Cards with errors are printed in CHECK as they are found.

The last Circuit Element Card contains the word NOMOR, which causes the program to either 1) print a reference list of error codes if Circuit Element Cards had errors, or 2) rearrange the circuit elements in numerically increasing order if all the Circuit Element Cards seemed to be correctly prepared. This rearranging is done in subroutine EXCH, which is entered for each circuit element. Circuit elements which were supplied with numbers are put in the position in the circuit element array which they would have had if their numbers were used as their index in the array. Circuit elements which do not have numbers are assigned numbers that are unfilled by other elements. These numbers reflect the order of the circuit elements in the data deck. The program will stop if two circuit elements have been assigned the same number.

The circuit elements are now printed in their new order by subroutine INOUT. We do not use this portion of the main program or INOUT, EXCH, and CHECK again, unless a NEXT card rather than a FINI card is used at the end of the output requests for this circuit.

1.2 FORMING THE EQUIVALENT CIRCUIT

An output request card, called a Type Card in the User's Manual, is read in to request either an AC or DC solution. Subroutine ACEQ will be called for AC, subroutine DCEQ for DC. Each of these subroutines shorts or opens certain types of circuit elements; tests the resulting circuit for connectivity (using subroutine CONECT); and condenses the resulting circuit so that the names of the nodes are sequential, beginning with zero and progressing through the maximum node. No node numbers are skipped.

1.3 ASSIGNING VALUES TO CIRCUIT ELEMENTS

In addition to denoting the type of circuit, the Type Card also specifies the type of solution required. The choices are: Nominal, Special, Sensitivity, Monte Carlo, or Frequency Plot. The Plot and Sensitivity output have two choices each,

and the Nominal and Special Solutions could also have the nonlinear option. Thus, some programming is required to assign proper values to the circuit elements for these different solutions.

The main program first tests to see if the solution requires nonlinear circuit elements. If so, each unknown node requires a starting value. These values, together with the upper and lower limits on solutions and their tolerances, are read in. Subroutine NOLIN is called to initialize the solutions. Next the program branches, depending on the type of solution required. We will consider each branch individually.

a. If the solution requested is Nominal, the program reads in the frequencies required (in the AC case) by calling subroutine READFQ. Next, the program reads any nodal, branch-current, and special-equation output requests. Finally, nominal values are inserted in the circuit elements of the equivalent circuit and the program transfers to statement 635, the beginning of the solution loop.

b. A Special solution request causes the program to read in the frequencies required (in the AC case) by calling READFQ. It then starts the loop on the number of special solutions this Type Card specified (columns 7-8) and calls subroutine SPECIN, where node output requests and special values for the circuit elements are read in. The appropriate special value is assigned to each circuit element and their values are printed out. The program returns to MAIN, sets an indicator (NO) to 5 to show a Special solution request, and goes to statement 635.

c. A Sensitivity solution request causes the program to read in a card specifying the mode of sensitivity analysis (see below); whether phase, magnitude, or both types of output are required (in the AC case); and the number of frequencies, up to five. Thus for DC Sensitivity requests, this card only contains the mode of the output. The main program then sets an indicator (NO) to one to indicate a Nominal request, and transfers to the Nominal part of the main program where nodal and current output requests are read in. The difference in the two modes does not become apparent until output has been accumulated, as described in Section 1.6.

d. A Monte Carlo solution request reads in a card giving the number of solutions desired for each frequency and the starting value for the random number generator. If the number of solutions requested exceeds 999, the next solution request card is read and the Monte Carlo solution is not done. As in the Sensitivity solution, the number of frequencies (up to five) and their values are read in on this card. Up to five Special Function output requests are read in next. Since only five output values are allowed, the usual sequence of node outputs, branch currents, and special function output selection cards does not apply. Instead the output requests are entered as special functions and are requested by entering equations in EQUOUT.

   The cards specifying the special functions desired for output also specify upper and lower bounds on the Histogram plot routine. Further, an indicator is set for each output so that either phase or magnitude may be requested. The program tests that the lower limit is lower than the upper limit. If these cards were improperly prepared, the program searches

for the next Type Card and skips this Monte Carlo request. If no high or low limits were set, the program uses ±20 percent of the nominal solution. Normally the nominal solution is not needed for a Monte Carlo request, but in this case it is, so we set NO = 1 and go to statement 635 to compute the nominal solution first.

If the upper and lower limits were supplied, then we begin supplying circuit element values in the following way: The starting value of the random number generator is the value read in on the card following the Type Cards. The first sample will use as many separately generated random variables as there are circuit elements in the equivalent circuit. (Admittance and impedance will require two random variables for their values.) Each circuit element is examined to discover its distribution. If the distribution was read in as zero on the Circuit Element Card, the program provides the nominal value to that element and goes on to the next one. If the distribution code was non-zero (i.e., from 1 to 4), a random variable is generated by subroutine RANDU, and the appropriate subroutine is called to transform the uniformly distributed random variable into a normal (NORM), lognormal (LOGNOR), rectangular (RECTAN), or special (SPEC) distribution. The value produced is assigned to the circuit element. Admittance and impedance elements receive two random variables and enter the appropriate subroutine twice. When all the circuit elements have been assigned values, NO is set to 6 and the program transfers to statement 635 and begins the solution.

e. A Frequency Plot request is used for an AC equivalent circuit only. It collects solutions for various frequencies and produces a plot of solutions versus frequency using a Stromberg-Carlson CRT display unit.

The card after the Type Card gives the mode of the plot desired; whether a phase or magnitude plot, or both, is required; and whether the frequency axis is to be logarithmic or linear. Next the frequencies are read in by subroutine READFQ. The functional output requests are read in, each containing the number of grid lines desired and the maximum and minimum values of the output. If we are in mode 1, the PLOTF subroutine computes the number of grid lines and the maximum and minimum values of the output, so the Functional Output Request Card will not contain the mode 2 information. The PLOTF can only be used with nominal values, so at this point, the first frequency is set and the control goes to the part of the program that supplies nominal values to each circuit element.

f. A Nonlinear solution for either NOML or SPCL output requests requires additional cards. Thus, immediately after the Type Card specifying a nonlinear circuit is processed, cards are read in giving 1) the node number of a node voltage referenced by an equation in EQUIN, the initial value of this voltage; and 2) if available, upper and lower bounds on the value of the node voltage. The tolerance can also be supplied. If upper and lower

bounds are not supplied, subroutine NOLIN sets these bounds to ±50 percent of the first guess. If the tolerance is not supplied, it is set at 0.2 percent of the guess, and if the tolerance supplied is less than 0.01 percent of the guess, it is set up to 0.2 percent. The program then proceeds with a solution as if it were an ordinary Special or Nominal request.

1.4 OBTAINING SOLUTIONS TO EQUIVALENT CIRCUIT

This section of the program is common to all the output selections, and is broken up into five subroutines to reflect the five steps necessary in obtaining a solution.

a. The main program calls subroutine ASSIGN to complete the assignment of values to and compute impedance for circuit elements. Here the value of NO directs the subroutine when it must find the imaginary parts of admittance and impedance elements. The ASSIGN subroutine also indicates current flow by setting the sign for active elements (V, E, D, I, B, H, G, and F types), and computes the values for circuit elements dependent on other elements by computing their value in an equation in EQUIN.

b. On return from ASSIGN, the main program immediately calls subroutine SOLUT. This subroutine prints the equivalent circuit, if NDEBUG on the Type Card of this output request was 1, and calls subroutine CURENT for V, E, H and G circuit elements. These elements are voltage sources, and since we are writing node equations when we form our simultaneous linear equations, our active circuit elements must be current. Subroutine CURENT performs the conversion from voltage to current by putting a resistor in parallel with the voltage in the circuit and dividing the voltage by the resistance. The value of the resistance is 1 ohm unless supplied by the engineer on the voltage circuit element card. The resistor is entered as a circuit element in the equivalent circuit, and this circuit is printed out again if NDEBUG = 1.

c. On return to the main program from SOLUT, the main program immediately calls subroutine COMPUT, where the node equations are formed. At the end of COMPUT, if NDEBUG = 1, the matrix formed is printed out. The values of the coefficients are printed out, not an algebraic representation of the equations.

d. Finally, COMPUT calls SOLVE, which solves the simultaneous linear equations formed by COMPUT. The technique employed is a modified Gauss elimination method in double precision. Real matrices are solved in a different portion of the subroutine from complex matrices. Because the hardware for double precision varies between machines, considerable effort was made to avoid unnecessary computation in SOLVE. This is discussed in Section 3 under subroutine SOLVE.

   On return to COMPUT from SOLVE, the solutions are put in the appropriate position in the OUTPUT array so the engineer's, rather than the program's, numbering of nodes prevails. Remember that on conversion to the equivalent circuit, the nodes were renumbered so that no nodes were skipped.

1.5 FURTHER PROCESSING OF VOLTAGE OUTPUT

On return to the main program from COMPUT, the program first tests to see if we had a singular (or ill-conditioned) matrix. This would indicate that no solution was obtainable with the values the engineer supplied. (It sometimes indicates that he has not supplied values in the "high" and "low" columns — second and third data fields — of the Circuit Element Card, while still specifying a sensitivity or special solution using those elements.) The program will allow two such "no solution" attempts before terminating the run.

a. Nonlinear Solution – Next, if the solution request was for a nonlinear circuit, subroutine NOLIN is called. NOLIN tests to see if this solution is adequate, or if new values for the unknown outputs are needed. In the first case, INDIC = 2 and we print out nominal or special solutions. If INDIC = 1, we must iterate the solution again, and we branch to the Nominal or Special solution sections of the program, whichever is appropriate. If the nominal values are needed, they are assigned to the circuit elements and the program returns to statement 635 which calls subroutine ASSIGN. If special values for the circuit elements are needed, the main program calls subroutine SPECIN and then goes to statement 635.

   If INDIC = 3, the iterations have been used up without obtaining a solution, so we print a diagnostic and search for the FINI or NEXT card.

   If INDIC = 2 or if we have a linear circuit, we branch on the kind of solution obtained as in the linear case.

b. Nominal or Special Soution – We take the same branch and write the title, type of solution, and solution subtitle; write the frequency (if this is an AC circuit); and call POLAR. This subroutine calls subroutine OUT, which computes magnitude and phase. Then, on return to POLAR, the outputs requested by the engineer are computed and printed. An expansion of the logic employed in POLAR may be found in Section 3.

   On return to the main program, we branch if the solution is a special one. For the Nominal solution of a DC equivalent circuit, we return to statement 252 and read the next Type Card. If this is an AC equivalent circuit, the frequency index test (LF) is incremented to see if we have just computed the last frequency (if so we go to statement 252), and return to the part of the program where nominal values are supplied to the circuit elements.

   For a Special solution of a DC equivalent circuit, we 1) print the title and call POLAR, 2) increment the counter, ISP, and 3) test to see if it exceeds NSP, the limit supplied by the engineer on the number of Special solutions he wanted. If there are still Special solutions to be read in and performed, subroutine SPECIN is called to read the cards giving the special values to be used. On return to the main program, we go to statement 635 and complete the solution. If we have read all the Special solution cards (when ISP is greater than NSP), we go to statement 252 to read the next Type Card.

For an AC Special solution request card, we 1) print the title and frequency, 2) call POLAR, 3) increment the frequency index, LF, and 4) test to see if the Special solution has been performed for all the frequencies requested (is LF greater than NLF?). If so, we increment the Special solution counter, ISP, and the program behaves as it did in the DC case, above.

c. Sensitivity – For this branch, we first test to see if we have just completed the Nominal solution (needed as a base for Sensitivity analysis) or if we are already varying the circuit elements. This is reflected by the value of NO. If NO = 1, the Nominal solution has been computed and is saved in COMP. The Sensitivity variation is started by initializing ISN, which controls the variation of the circuit element values. If the first element is not to be varied in a Sensitivity analysis, NSENSE (1) will be zero. In this case we increment ISN by one, test to see that we have more circuit elements in the input circuit (engineer supplied, not the equivalent circuit), and continue testing the NSENSE array until a circuit element is found that should be varied. When such an element is found, we set all other values to nominal and set the one circuit element we are varying to its low value (columns 41-50 of the element's card); set NO to 2; and go to statement 635 to continue the solution.

If the tested value of NO is 2 (this is on return from finding a low value solution), we store the low solution's node voltages in the upper half of the COMP array (the lower half contains the nominal voltages) and put the high values in the ISN circuit element being varied, with the remaining elements being reset to their nominal value. NO is set to 3 and we go to statement 635 to obtain a solution.

If the tested value of NO is 3, we transfer to a test to determine the mode of sensitivity the engineer requested for output. If he requested mode 1 (the mode which simply prints the outputs requested and doesn't order them or do a worst case analysis), we print the title if this is the first circuit element varied, print the frequency if this is an AC circuit, and then call subroutine SENSPR, which computes and prints the outputs requested. On return from SENSPR, ISN is incremented and tested to see that all circuit elements have been varied. If there are still circuit elements to be varied, we transfer to the part of the program where NO is set to 2 and put the low values in the next element that the engineer wished to be varied. If all circuit elements have been varied (ISN is greater than JCOMP after being incremented), NO is set to 1, and we test to see if we are on mode 1 or mode 2 Sensitivity analysis; since we are on mode 1, we test to see whether an AC or a DC circuit is being solved. If we have just finished varying a DC circuit, we go to statement 252 and read the next Type Card. For an AC equivalent circuit, the frequency index is incremented. If the circuit has not been solved for all requested frequencies, we return to statement 611 and begin a Nominal solution for the new frequency.

For a mode 2 Sensitivity solution (after solving the circuit for a high value), subroutine SENSP1 is called. This subroutine stores outputs until all circuit elements have been varied. On return from SENSP1, ISN is incremented and tested to see if all circuit elements have been varied. If not, we transfer to the part of the program where NO is set to 2 and begin on the solution of the circuit with a new circuit element set to its low value.

If the circuit has been solved for all low and high value combinations requested, we then test to see if we are on mode 1 or mode 2. Since a mode 2 branch is now being examined, we test to see if a worst case analysis (low or high) has been solved. This is indicated when NEXIT = 1 or 2. If NEXIT = 0 we have not stored the worst-case low values of the circuit elements so we must obtain a worst case solution. We also must order and print the preceding variations. For this purpose, SENSP1 is called again. On return from SENSP1, if NEXIT = 1, ISN is still greater than JCOMP, the mode is still 2, and NEXIT is neither 0 nor 5, then we set NO = 4 (to help ASSIGN find the correct admittance and impedance values) and transfer to statement 635, where we begin a solution of the worst-case low variation.

On return from COMPUT, we take the Sensitivity solution branch, test that NO is greater than 2 and that the mode is 2, and call SENSP1 again. SENSP1 stores the low worst-case solution, computes the high worst-case values for the circuit elements, sets NEXIT to 2, and returns to the main program. ISN is still greater than JCOMP, and NEXIT is now 2, so we eliminate the previous tests and go to statement 635 where the solution begins. On return to SENSP1, the worst case high and low solutions are printed out and SENSP1 tests to see if there are more output requests to vary in a worst case analysis. (Remember that worst case combinations of circuit elements are dependent on the sign of the output when we subtract high solutions from low solutions. Thus we must compute separate worst cases for each output requested.) If all outputs have not been considered, the next output values are ordered and printed out, worst-case low values are computed for the components, NEXIT is set to 1, and we return to the main program. If all the outputs have been computed, NEXIT is set to 5 and on return to main we either read the next Type Card (DC case) or vary the frequency (AC case).

d. <u>Frequency Plot Solution</u> – On return to the main program from COMPUT, we branch to a part of the program which calls the PLOTF subroutine. PLOTF stores the outputs obtained, but doesn't do any plotting until LF = NLF, where LF is the index of the present frequency and NLF is the index of the last frequency to be computed. On return to the main program from PLOTF, the program increments LF by one, and tests to see if all NLF frequencies have been computed. If not, we go to statement 611 where the nominal values of the circuit elements in the equivalent circuit are supplied. If all NLF frequencies are computed, then the last time PLOTF was called a plot of the outputs was produced, so we are through with this output selection. The program transfers to statement 252 and reads a new Type Card.

e. <u>Monte Carlo Solution</u> – On return to the main program from COMPUT, we branch to a part of the program which tests if NO = 1. This indicator is set to 1 when we have a Nominal solution, and the only time a Nominal solution would be needed in a Monte Carlo analysis would be when we didn't supply high and low limits on the histogram requested as the output for the Monte Carlo analysis. So when NO = 1 we must transfer back to the part of the program which initializes a Monte Carlo solution. Here (statement 483), we compute the nominal values of those outputs which do not have high and low limits and then set the high and low limits to ±20 percent

of the nominal value. Then we begin the Monte Carlo initialization as described in the beginning of this section.

If NO is not 1, we call subroutine STAT, which stores the solutions for later printing in HIST, the histogram plot routine. On return from STAT, we transfer to an earlier portion of the program where IR is incremented. IR is the counter that keeps track of the number of samples taken so far. Each IR generates a new solution of the circuit with new values supplied to the circuit elements which the engineer wanted to vary randomly. After incrementing IR, the program tests to see if we have as many samples as the engineer requested (NORV). If IR is not greater than NORV, we transfer to the part of the program which supplies new values to the circuit elements.

If we have completed NORV samples, we go to statement 252 and read a new Type Card.

2. CROSS-REFERENCE OF ARRAYS

Variable	Found in
A	CHECK, ASSIGN, SPCE, COMPUT, SOLVE, EXCH, RECTAN, INOUT, MAIN
AC	SPECIN, POLAR, MAIN
ADMIT	CHECK, ASSIGN, MAIN
ALL	SPECIN, MAIN
AST1	HIST
B	CHECK, ASSIGN, EXCH, RECTAN, PLOTF, SOLUT, CURENT, COMPUT, MAIN
BEG	STAT
BEND	PLOTF
BIN	STAT
BLANK	READFQ
BLANK1	CHECK, MAIN
BLANK4	CHECK, MAIN
BVAL	COMPUT
C	CHECK, ASSIGN, DCEQ, MAIN
CARD	EQUOUT, CARD, EQUIN
CARLO	CHECK, INOUT, MAIN
CHECK	READFQ
COL8L	HIST
COL8R	HIST
COLMID	HIST
COMP	SPECIN, SENSP1, BRANCH, POLAR, MAIN
CUR	SENSP1
CURR	CURR, EQUOUT, EQUIN
D	CHECK, ASSIGN, DCEQ, COMPUT, ACEQ, MAIN
DC	SPECIN, SENSP1, MAIN

Variable	Found in
DEL	PLOTF, NOLIN
DELT	NOLIN
DEN	ASSIGN
DIAG	DCEQ, CURRENT, ACEQ, MAIN
DIF	SPCE, STAT, NOLIN, NORM
DIST	SPCE
DIV	COMPUT, BRANCH, SOLVE
DIVI	SOLVE
DIVR	SOLVE
DSAVE	NOLIN
DUM	ASSIGN, SOLUT, CONECT, EQUIN
DUMB	READFQ
DX	PLOTF
E	CHECK, MAIN, ASSIGN, EXCH, SOLUT, DCEQ, CURENT, COMPUT
ER	PLOTF, EQUOUT
ERR	EQUIN
ERROR	ASSIGN
F	CHECK, MAIN, ASSIGN, EXCH, PLOTF
F13	STAT
F25	HIST
FINI	MAIN
FINISH	STAT
FREQ	MAIN, ASSIGN, PLOTF, STAT, SENSPR, POLAR, EQUIN, READFQ, SENSP1, EQUOUT
FS	STAT
G	CHECK, MAIN, ASSIGN, EXCH, CURENT
GI	MAIN
GSAVE	NOLIN
GUESHI	MAIN, NOLIN

Variable	Found in
GUESLO	MAIN, NOLIN
GUESS	NOLIN
H	CHECK, MAIN, ASSIGN, EXCH, SOLUT, CURENT, COMPUT
HI	MAIN
HIHIST	MAIN, STAT
I	OUT, EXCH, PLOTF, CURENT, READFQ
I1	SOLUT, DCEQ, SENSPR, SOLVE, ACEQ
I2	SOLVE
IAD	CHECK, MAIN, ASSIGN, EXCH, INOUT
IC	CHECK, MAIN, EXCH, INOUT
ICI	CURENT
ICOM	SENSP1
ICOMP	CARD, MAIN, ASSIGN, SPECIN, SOLUT, DCEQ, CURENT, COMPUT, BRANCH, POLAR, SENSP1, CURR, ACEQ
ICR	POLAR
IE	CHECK, MAIN, ASSIGN, EXCH, INOUT
II	COMPUT, SOLVE
IK	STAT
IL	MAIN
IM	STAT, COMPUT
IN	MAIN
INDIC	MAIN, NOLIN
INL	MAIN
INODE	MAIN, ASSIGN, SOLUT, CONECT, DCEQ, CURENT, COMPUT, POLAR, SENSP1, CURR, ACEQ
INPUT	EQUIN
IOUT	DCEQ
IOVER	SPCE

Variable	Found in
IP	CHECK, MAIN, EXCH, SOLUT, CURENT
IPOW	DCEQ
IPR	SPECIN, SENSPR, POLAR
IPR1	SENSPR
IPR2	SENSPR
IR	MAIN, STAT, COMPUT, SOLVE
IS	CHECK, MAIN, ASSIGN, EXCH, INOUT, SPECIN, SENSP1
ISN	MAIN, SENSPR, SENSP1
ISP	MAIN, SPECIN
IST	NORM
ISTAT	MAIN, STAT
IT	CHECK, MAIN, EXCH, INOUT, STAT, COMPUT
IT123	READFQ
ITI	CURENT
ITYPE	MAIN, ASSIGN, SOLUT, DCEQ, CURENT, COMPUT, ACEQ
IV	CURENT
IVALUE	MAIN, SPECIN, SOLUT, DCEQ, CURENT, BRANCH, SENSP1, CURR, ACEQ
IW	MAIN, STAT
IWHICH	MAIN, PLOTF, SENSPR, SENSP1
J	CHECK, EXCH, BRANCH, READFQ, SENSP1, HIST
J2	PLOTF, SOLVE
J3	STAT
JB	COMPUT
JBOX	STAT
JC	MAIN, DCEQ, CURENT, SENSP1, ACEQ, CURR
JCOMP	MAIN, INOUT, SPECIN, SOLUT, CURENT, POLAR, SENSP1
JI	SOLVE
JNODE	MAIN, SOLUT, DCEQ, CURENT, COMPUT, ACEQ

Variable	Found in
JR	SOLVE
JS	SPECIN, PLOTF, STAT, SENSPR, POLAR, SENSP1
JS1	MAIN, SENSPR
JSC	CURENT
JT	STAT
K	CHECK, MAIN, ASSIGN, PLOTF, STAT, DCEQ, NOLIN, SENSPR, COMPUT, SOLVE, POLAR, READFQ, SENSP1, HIST, ACEQ
K1	SPECIN, DCEQ, CURENT, SOLVE
K2	SPECIN, DCEQ, CURENT, SOLVE, HIST
K22	CURENT
KB	STAT, COMPUT
KBR	COMPUT
KCOMP	CURENT
KCT	HIST
KE	ASSIGN
KE1	ASSIGN
KI	COMPUT, SOLVE
KM	STAT
KR	ASSIGN, COMPUT, SOLVE
KS	MAIN, STAT, SOLUT, CURENT
KSEN1	MAIN
KSEN2	MAIN
KV	MAIN, SPECIN
KV1	CURENT
KV2	CURENT
L	ASSIGN, CHECK
L1	HIST
LABEL	HIST, STAT
LB4	HIST
LEQ	SENSP1, PLOTF

Variable	Found in
LEQU	MAIN, HIST, SENSP1, POLAR, SENSPR, STAT, PLOTF, SPECIN
LF	MAIN, SENSP1, POLAR, SOLUT, SPECIN
LG	MAIN
LIG3	STAT
LIM	NOLIN
LIMIT	NOLIN
LL	READFQ
LPRNT	HIST
LTYPE	READFQ
M	SPECIN, PLOTF, STAT, DCEQ, ACEQ, SENSP1, ASSIGN
M2	STAT
MAX	HIST
MB	COMPUT
MBR	COMPUT
MC	MAIN
MD	SPCE
MER	CHECK
MIN4	HIST
MINMID	HIST
MODE	PLOTF, MAIN
MONT	MAIN
MS	SOLUT, MAIN
N	EQUIN, EQUOUT, CONECT, CURR, ACEQ, SENSP1, SOLVE, SPCE, PLOTF, COMPUT, DCEQ, ASSIGN
N1	SOLVE
N2	SOLVE, EXCH
N200	MAIN
NAC	MAIN, NOLIN
NAD	SOLVE
NALL	MAIN

Variable	Found in
NAMBAD	CHECK
NAME	SENSPR, SPECIN, POLAR, MAIN, SOLUT, SENSP1, INOUT, EXCH, CHECK
NAMCP	SENSP1
NAMO	SENSP1
NARLO	MAIN, EXCH, CHECK
NB	CURENT, SPECIN, CONECT, STAT, SOLUT, PLOTF, COMPUT, DCEQ
NBETWH	STAT
NBETWL	STAT
NBOX	STAT
NBR	SPECIN
NBRAN	MAIN
NC	SPECIN, POLAR, STAT, MAIN, SENSP1, SOLUT, NOLIN, COMPUT
NCLOS	NOLIN
NCOMP	CURENT, CARD, CURR, MAIN, INOUT, COMPUT, EXCH, DCEQ, CHECK
NCON	CURR, MAIN
NCR	MAIN
NCUR	SPECIN, POLAR, SENSP1
ND	CURENT, CONECT, ACEQ, MAIN, SOLUT, NOLIN, COMPUT, DCEQ
ND1	CURENT, CONECT, ACEQ, SOLUT, COMPUT, DCEQ
ND2	COMPUT
NDC	MAIN, COMPUT, ASSIGN
NDEBUG	MAIN, SOLUT, NOLIN, COMPUT
NDIF	SOLVE, READFQ
NDIM	SOLVE
NDIST	MAIN, INOUT, EXCH, CHECK
NDMIT	MAIN, INOUT, EXCH, ASSIGN, CHECK
NDR	CONECT, ACEQ, DCEQ
NE1	BRANCH
NE2	BRANCH

Variable	Found in
NEG	NORM
NEQU	SENSPR, CURENT, BRANCH, POLAR, CURR, ACEQ, MAIN, SENSP1, SOLUT, NOLIN, COMPUT, DCEQ, ASSIGN
NEQUAT	MAIN, INOUT, EXCH, ASSIGN, CHECK
NER	MAIN, COMPUT, CHECK
NERROR	MAIN, SOLUT, CHECK
NEW	HIST
NEXIT	MAIN, SENSP1
NEXT	MAIN
NF	CURENT, CONECT, STAT, SOLUT, PLOTF, COMPUT, DCEQ
NF1	STAT
NFG	PLOTF
NFGRID	MAIN, PLOTF
NFIRST	CONECT
NGRID	MAIN
NHOLD	SPECIN, MAIN
NIM	MAIN
NINT	PLOTF
NL	SPECIN, MAIN
NLE	PLOTF
NLF	READFQ, MAIN
NLF2	READFQ
NLF3	READFQ
NLFSP	READFQ
NM1	STAT, SOLVE
NM2	STAT
NMC	MAIN, INOUT, EXCH, CHECK
NMC1	CHECK
NND	ACEQ, DCEQ
NO	CONECT, ACEQ, MAIN, SENSP1, ASSIGN
NOD	SENSPR, MAIN, SENSP1

Variable	Found in
NODE	SENSPR, BRANCH, POLAR, CURR, ACEQ, MAIN, SENSP1, SOLUT, INOUT, COMPUT, EXCH, DCEQ, ASSIGN, CHECK
NODMAX	SENSPR, BRANCH, SPECIN, POLAR, ACEQ, MAIN, SENSP1, SOLUT, DCEQ
NODMX1	ACEQ, MAIN, SOLUT, COMPUT, DCEQ, ASSIGN
NODNAM	MAIN, NOLIN
NOML	MAIN
NOLIN	MAIN, NOLIN
NORV	STAT, MAIN
NOTCON	CONECT, ACEQ, DCEQ
NOUT	SENSPR, SPECIN, POLAR, MAIN, SENSP1
NOUND	STAT
NOUNDHI	STAT
NOUNDLO	STAT
NOW	CONECT, STAT
NPEC	SPECIN, MAIN, EXCH, ASSIGN, CHECK
NPOW	CURENT, MAIN, SOLUT, EXCH, CHECK
NPW	MAIN, CHECK
NR	STAT, CHECK
NRDC	CHECK
NRDS	CHECK
NRM	STAT
NRT	STAT
NS	CONECT, ACEQ
NS1	CONECT
NSAV	READFQ
NSAVE	CURENT, CONECT, ACEQ, DCEQ
NSCR	CURENT, CURR, ACEQ, MAIN, SOLUT, DCEQ
NSENSE	MAIN, INOUT, EXCH, CHECK
NSP	MAIN

Variable	Found in
NSPEC	SPECIN, MAIN, INOUT, EXCH, CHECK
NST	SPECIN, MAIN
NSUM	STAT
NSUT	SPECIN, MAIN
NT	STAT, EXCH
NT1	STAT
NTOP	NORM
NTYPE	SENSPR, SPECIN, POLAR, ACEQ, MAIN, SENSP1, SOLUT, INOUT, EXCH, DCEQ, CHECK
NUM	SENSPR, MAIN, SENSP1, CHECK
NV	SENSPR, SPECIN, MAIN
NW	SENSP1
NWC	SENSP1
NWORST	SENSP1
NX	PLOTF
NX2	PLOTF
NXT	HIST
NY	PLOTF
NY2	PLOTF
NYGRID	PLOTF
NZ	MAIN
OU	MAIN, CURR, BRANCH, POLAR, SENSPR, PLOTF
OUT	COMPUT
OUTPUT	MAIN, ASSIGN, CURR, BRANCH, POLAR, SENSP1, SENSPR, OUT, EQUOUT, PLOTF, STAT, NOLIN
PI	SOLVE
PI1	SOLVE
PI2	SOLVE
PLOT	MAIN
PLTLE	MAIN
PNOM	SENSP1

Variable	Found in
PR	SOLVE
PR1	SOLVE
PR2	SOLVE
R	SOLUT, COMPUT, CURENT, ASSIGN, CHECK, MAIN
RANGE	STAT, PLOTF
RCT	HIST
RE	MAIN, OUTPUT, COMPUT
RI	SPCE
RN	STAT
RN1	STAT
RR	SPCE
RV	SPCE, NORM, RECTAN, MAIN
RV2	MAIN
RVAL	COMPUT
SENS	MAIN
SIG	HIST, NORM, STAT
SIGNO	STAT
SPCL	MAIN
SPEC	MAIN, CHECK, ASSIGN, INOUT, SPECIN
SQUAR	STAT
STAN	STAT
START	MAIN, READFQ
STEP	READFQ
STORE	MAIN, SENSP1, PLOTF
STR	STAT
SUBTLE	MAIN, SENSP1, SPECIN, STAT, SOLUT
SUM	SOLVE
SUMDIF	NOLIN
SUMI	SOLVE
SUMR	SOLVE

Variable	Found in
SUMTOL	NOLIN
SV	MAIN, SENSPR, SPECIN
TENPC	PLOTF
THIS	STAT
TITLE	MAIN, STAT, SPECIN, SENSP1
TOL	MAIN, NOLIN
TRANS	MAIN, EXCH, INOUT, COMPUT, CHECK
TRY	NOLIN
TTLPT	PLOTF
TYPE	MAIN, SOLUT, STAT, SPECIN, SENSP1
V	CHECK, ASSIGN, ACEQ, CURENT, DCEQ, COMPUT, EXCH, SOLUT, MAIN
VAL	RECTAN, NORM, CONECT, SPCE, MAIN
VAL2	MAIN
VALUE	ASSIGN, COMPUT
VARI	STAT
VI	SPCE
VR	SPCE
VOLT	EQUIN, CURENT, EQUOUT
VOLT2	CURENT
W	ASSIGN
WCMAXM	SENSP1
WCMAXP	SENSP1
WCN	SENSP1
WCSAVIN	SENSP1
WCSAVX	SENSP1
WCX	SENSP1

Variable	Found in
X	NOLIN
XF	PLOTF
XHIGH	CHECK, ASSIGN, SENSP1, SENSPR, INOUT, EXCH, SPECIN
XIM	OUT
XINT	STAT
XLHIST	MAIN, STAT
XLOW	CHECK, MAIN, ASSIGN, SENSP1, SENSPR, INOUT, EXCH, SPECIN
XM	NORM, SPCE
XMAG	MAIN, POLAR, SENSP1, PLOTF, OUT
XMAX	SOLVE, PLOTF, MAIN
XMCHI	CHECK, MAIN, ASSIGN, INOUT, EXCH, SPECIN
XMCLO	CHECK, MAIN, ASSIGN, INOUT, EXCH, SPECIN
XMEAN	NORM, STAT
XMED	STAT
XMIN	SOLVE, PLOTF
XN	EXCH
XNINT	PLOTF
XNOM	CHECK, MAIN, ASSIGN, SENSP1, CURENT, INOUT, EXCH, SPECIN, SOLUT
XNORM	NORM
XP	STAT
XPHS	MAIN, POLAR, SENSP1, PLOTF, OUT
XR	CURENT
Y	NOLIN, COMPUT, DCEQ, ASSIGN, CHECK
YD	PLOTF
YL	HIST
Z	INOUT, SPCE, EXCH, ASSIGN, CHECK

3. SUBROUTINES

ACEQ (NODE, NTYPE, ITYPE, INODE, ND, V, D, NODMAX, NEQU, DIAG, JNODE)

ACEQ is called by the main program whenever an AC circuit solution is requested and either 1) this is the first solution request after the Circuit Element Cards are read in, or 2) the previous solution request was for a DC circuit. ACEQ converts the read-in circuit to an AC equivalent circuit by opening DC current sources and shorting DC voltage sources. In the case of voltage, ACEQ accomplishes this by bringing the two nodes on each side of the voltage source "together" so that a node is lost. The dropped node is renumbered to correspond with the retained node, and all references to it in the read-in circuit are also renumbered. Any circuit elements in parallel with the shorted voltage source necessarily become shorted.

Current sources are opened by dropping any circuit elements of type D from the equivalent circuit being formed. Because the network has been opened, it may now be unconnected, so subroutine CONECT is called to test this. If the circuit is now without power in all its subcircuits, it is an incorrect equivalent circuit. A flag is set to indicate this and an error return is taken to the main program.

Otherwise (if the network is powered), the equivalent circuit is constructed from the scratch array, NSCR, in which the nodes in the equivalent circuit are stored. Forming the equivalent circuit consists of going through the scratch array and finding which nodes are connected to each other. ACEQ also, at this time in the program, eliminates any node numbers that do not have links coming from them. This compresses the network and makes the "table of contents" array, NEQU, necessary.

Finally, the program tests that at least one node is named "0" for ground, i.e., that "0" was not an eliminated node; and then returns to the main program.

D	DC circuit element.
DIAG	0, usually. If subroutine CONECT finds the circuit separate when it shouldn't be, DIAG = 1.
I1	I + 1, necessary when examining arrays concerning nodes, since ground is zero.
ICOMP (400)	An array of circuit element numbers which serve as an index between the read-in circuit and the equivalent circuit.
INODE (1)	One of the nodes between which a circuit element in the equivalent circuit lies.
ITYPE (1)	The circuit element type in the equivalent circuit, either R, L, C, V, E, D, I, A, Z, B, H, F, or G.
IVALUE (2,400)	A place-saver in the COMMON listing.
JC	The number of circuit elements times two in the equivalent circuit.
JNODE (2, 1)	An array that serves as a table of contents for the array describing the equivalent circuit. See appendix on Network Storage.

ACEQ (Continued)

K	Temporary storage for a particular value of ICOMP. This is necessary since the entries of ICOMP are themselves used as indices.
M	Temporary storage of a node in the read-in circuit.
N	Temporary storage of a node in the read-in circuit.
ND	The maximum node in the equivalent circuit.
ND1	The number of nodes in the equivalent circuit plus one (to account for ground being zero).
NDR	A counter of the number of nodes with links in the equivalent circuit. Since some nodes may be skipped at this point in the program, NDR may be less than ND, the maximum node in the equivalent circuit.
NEQU (1)	An array of node name equivalences between the read-in circuit (contained in the index + 1 of NEQU) and the node number of the equivalent circuit. NEQU (5) = 3 means that node 4 of the read-in circuit is numbered node 3 of the equivalent circuit.
NND	Temporary storage for ND + 1 when the compressed equivalent circuit is being constructed.
NO	The number of nodes in the NOTCON array.
NODE (2, 1)	The node numbers between which this component lies. NODE (1, J) is the primary node, NODE (2, J) is the secondary node.
NODMAX	The maximum node in the read-in circuit.
NODMX1	The maximum value of the nodes plus one in the read-in circuit.
NOTCON (51)	An array of nodes which, after return from subroutine CONECT, the program has found unconnected to the nodes in NSAVE.
NS	The number of nodes in the NSAVE array.
NSAVE (51)	An array of connected nodes, as distinct from NOTCON, in the newly generated equivalent circuit.
NSCR (2, 201)	An array identical with the NODE array (of the read-in circuit) at the beginning of this subroutine. Gradually NCSR is transformed into an array containing the node linkages of the new equivalent circuit.

ACEQ (Continued)

NTYPE (1)	The circuit element type, either R, C, L, A, Z, E, V, D, I, B, H, F or G.
V	A DC voltage circuit element.

ACEQ calls subroutine CONECT.

ASSIGN (REQUAT, IE, IAD, ADMIT, ITYPE, A, Z, V, Y, E, D, INODE, NEQU, XNOM, NODE, NDC, R, C, L. B, FREQ, ERROR, NODMX1, NO, NDMIT, IS, SPEC, NPEC, OUTPUT, H, F, G, XLOW, XHIGH, XMCHI, NMCLO)

ASSIGN is called by the main program for any solution requested. On entry to ASSIGN, the values in the IVALUE array are the values supplied by the main program from the Circuit Element Cards (or from random number calculation in the case of a Monte Carlo solution), but are not necessarily in the form of impedance as they will be on exit from ASSIGN. After testing to see if the circuit element's value is supplied by an equation (in which case the impedance for that element is not calculated until elements not part of the equivalent circuit have been computed), ASSIGN tests to see if the element is an A or Z type. If it is, ASSIGN must supply the imaginary part of the value from the ADMIT array. If it is not a nominal solution, ASSIGN must find which value in the ADMIT array to use for the imaginary part. It does this by a combination of being directed by NO and by the value already in the real part of VALUE.

Next, ASSIGN tests to see if we have a power-source circuit element. If so, the sign of the value must reflect current flow through the element. If the circuit is an AC one, then C and L circuit elements must have a frequency multiplier for computation of their impedance.

Finally, if the value of the circuit element depends on an equation in EQUIN, ASSIGN computes its value; and, if it is a power source, sets its sign correctly.

A	Admittance-type circuit element.
ADMIT (1, 1)	Array containing the imaginary parts of the A and Z circuit elements. ADMIT (1, I) = circuit element number; ADMIT (2, I) = nominal imaginary value, etc.
B	Voltage-dependent current source.
C	Capacitor
D	DC current source.
DEN	Reciprocal of the magnitude of a complex number.
DUM	Place holder in the COMMON array.
E	AC voltage source.
ERROR	A dummy not used by ASSIGN.
F	Current-depencent current source.
FREQ	Frequency.
G	Current-dependent voltage source.
H	Voltage-dependent, voltage source.
IAD	Index controlling ADMIT.

ASSIGN (Continued)

ICOMP	Array of circuit element numbers for those elements in the equivalent circuit.
IE	Index controlling NEQUAT.
INODE (1)	Node in the equivalent circuit that the Ith circuit element lies between.
IS	Index used by the NPEC and SPEC array.
ITYPE (1)	Type of circuit element in the equivalent circuit.
K	Value of ICOMP for this circuit element, i.e., the circuit element number.
KE	Second node number plus 1 on the read-in circuit.
KE1	Actual KE node number.
KR	Second node number plus 1 in the read-in circuit.
L	Inductor-type circuit element.
M	Second index in the ADMIT array.
N	Value of NO used to find the right value in the ADMIT array.
NDC	0 for an AC circuit, 1 for a DC circuit.
NDMIT (6, 1)	Array containing the imaginary parts of the A and Z circuit elements. ADMIT (1, I) = circuit element number, ADMIT (2, I) = nominal imaginary value, etc.
NEQU (1)	Array of node equivalences between nodes in the read-in circuit and nodes in the equivalent circuit.
NEQUAT (2, 1)	Array containing circuit element numbers in NEQUAT (1, I) and equation numbers in EQUIN in NEQUAT (2, I).
NO	A variable directing ASSIGN as to the type of solution requested. 1 means nominal values are to be put in value of the circuit element. 2 means low values for a sensitivity variation on ISN. 3 means high values for a sensitivity variation on ISN. 4 means worst case analysis and we must test real part to get high. 5 means special analysis. Test real part to obtain correct value. 6 means Monte Carlo analysis. Imaginary part already provided.
NODE (2, 1)	The primary (NODE (1, I)) and secondary (NODE (2, I)) between which the Ith circuit element in the read-in circuit lies.
NODMX1	Number of nodes plus 1 in the read-in circuit.

ASSIGN (Continued)

NPEC (6, 1)	SPEC array in fixed format, used to hold the circuit element's number in NPEC (1, I).
OUTPUT (2, 1)	Node voltage output required by subroutine EQUIN for nonlinear solutions.
R	Resistor.
SPEC (6, 1)	NPEC array holding special values for some of the circuit elements.
V	A DC voltage circuit element.
VALUE (2, 1)	Impedance of the Ith circuit element.
W	2π times frequency.
XHIGH (1)	High value of the circuit element for sensitivity analysis or a special value.
XLOW (1)	Low value of the circuit element for sensitivity analysis or a special value.
XMCHI (1)	Standard deviation of the Monte Carlo distribution (if a normal or lognormal distribution is required), or a special value.
XMCLO (1)	Mean of the Monte Carlo distribution (if a normal or lognormal distribution is required), or a special value.
XNOM (1)	Nominal values of the circuit elements.
Y	AC current source.
Z	Impedance-type circuit element.

ASSIGN calls subroutine EQUIN.

BRANCH (J, NEQU, OUTPUT, OU, NODE)

BRANCH is called by subprograms POLAR, SENPR, SENSP1, and CURR. It computes the branch current through circuit element J, the first argument. BRANCH first finds the nodes between which the circuit element lies, makes sure they have not been dropped on conversion to the equivalent circuit, and then computes the voltage between them. Next, BRANCH finds the impedance of the circuit element, divides it into the voltage to obtain the current, and returns to the calling program.

COMP (201)	Place-saver for BRANCH's COMMON statement.
DIV	Magnitude squared of the impedance. Later, DIV is the reciprocal of the impedance squared.
ICOMP (400)	The array of circuit element numbers in the equivalent circuit.
IVALUE (2,400)	Impedance of the circuit elements in the equivalent circuit. IVALUE (1, I) contains the real part, IVALUE (2, I) contains the imaginary part.
J	Circuit element number of the elements whose current has been computed.
NE1	The circuit element lies between two nodes, with NE1 being the first node on the Circuit Element Card.
NE2	The second node on the Circuit Element Card.
NEQU (1)	An array of node equivalences. If node numbers were changed on conversion to the equivalent circuit, say node 7 was changed to node 4, then NEQU (8) = 4.
NODE (2, 1)	The primary (NODE (1, I)) and secondary (NODE (2, I)) nodes that a circuit element lies between in the read-in circuit.
NODMAX	The maximum number of nodes in the read-in circuit.
OU (1)	Temporary storage of the voltage, later converted to current before return to the calling program.
OUTPUT (2, 1)	Contains the node voltages. OUTPUT (1, 3) is the real part and OUTPUT (2, 3) the imaginary part of node 2 output voltage.

BRANCH calls no subroutines.

CARD (M)

CARD is called by EQUOUT or EQUIN when the impedance of a particular circuit element is needed. Subroutine CARD finds the entry in the equivalent circuit and returns to the calling program.

CARD	The value of circuit element M.
ICOMP (400)	An array of circuit element numbers in the equivalent circuit.
NCOMP (201)	A place holder in this subroutine.

CARD calls no subroutines.

CHECK (J, NODE, NTYPE, NUM, IE, NEQUAT, IT, TRANS, XLOW, XHIGH, XMCLO, XMCHI, B, K, C, Z, A, L, V, Y, E, D, NERROR, NAME, NAMBAD, NSENSE, NDIST, XNOM, NSPEC, NMC, BLANK4, IAD, ADMIT, BLANK1, IC, NARLO, CARLO, IS, NPEC, R, NPW, NPOW, IP, H, F, G)

CHECK is called by the main program once for each component card read in. This subroutine makes eight checks on the accuracy of the circuit element's entries and reads in any additional cards required by this element. The eight checks are:

1. Are the node numbers greater than 50?

2. If the circuit element is a B, H, F, or G type, does it have a circuit element reference?

3. Is the circuit element type a recognized one?

4. If this is an A or Z circuit element, is there a follow-on card giving the imaginary parts of its values?

5. Is the Monte Carlo distribution specified as greater than 4?

6. If the Monte Carlo distribution is 1, 2, or 3, is the mean and standard deviation (or if 3, the upper and lower bounds) supplied?

7. If the distribution is a 4, are follow-on cards in the data deck?

8. If the distribution is a 4, is the distribution provided a cumulative one and is the ordinate scale normalized between 0 and 1?

A	Admittance-type circuit element.
ADMIT (6, 50)	Array used to hold the imaginary parts of A and Z circuit elements. ADMIT (1, I) is called NDMIT and holds the element's number.
B	Voltage-dependent current source.
BLANK 1	A word containing one blank used for comparison when we are checking the follow-on card of an A or Z circuit element.
BLANK 4	A word containing four blanks used for comparison on the A and Z follow-on cards, NAME array.
C	Capacitor.
CARLO (41, 30)	Same as NARLO, but with a floating point name.
D	DC current circuit element.
E	AC voltage circuit element.
F	Current-dependent current source circuit element.

CHECK (Continued)

G	Current-dependent voltage-source circuit element.
H	Voltage-dependent voltage source circuit element.
IAD	Index of the ADMIT array, used for imaginary parts of A and Z circuit elements.
IC	Index of the CARLO (or NARLO array).
IE	Index of the NEQUAT array which holds the circuit element number and its equation number.
IP	Index used by the NPOW array.
IS	Index used by the NPEC and SPEC array.
IT	Index of the TRANS array which holds the circuit element number of a B, H, F, or G element and its associated reference number.
J	The number of a circuit element in the data deck. If this is the third element read in, J = 3. Comes from the main program.
K	Available dummy used in CHECK.
L	Inductor-type circuit element.
MER (10)	Array used to flag particular types of circuit element errors.
NAME (201)	Up to 4 alphanumeric characters used to describe the circuit element, besides its type.
NAMBAD (100, 2)	An error array no longer used by CHECK.
NARLO (41, 30)	Array holding pairs of points describing a special distribution required by the circuit element whose number is in NARLO (1, I).
NCOMP (201)	Circuit element's unique number assigned by the engineer, or, if blank on the Circuit Element Card, assigned by CHECK to be 200 + J.
NDIST (201)	Array of either 0, 1, 2, 3 or 4 giving the codes for the Monte Carlo distributions of the circuit elements.
NDMIT (6, 50)	Fixed point array of ADMIT, not used in CHECK.
NEQUAT (40, 30)	Array of circuit element numbers (in NEQUAT (1, I)) and their associated equation numbers (in NEQUAT (2, I)) if their value must be computed. NEQUAT (3, I) is 0, 1 or 2 depending on the number of equation-labeling cards following the Circuit Element Card. Positions NEQUAT (4, I) through NEQUAT (38, I) hold the equation image for printout in INOUT.

CHECK (Continued)

NER (10)	Array used for printing out the MER array when errors are found.
NERROR	Error indicator, set to 1 by CHECK if any errors were found on the Circuit Element Cards.
NMC (201)	Array which tells the program, if a circuit element has special distribution for Monte Carlo analysis, how many points are in the distribution.
NMC1	Index computed from NMC. The number of values read into the NARLO array is twice the number of points in the special distribution.
NODE (2, 201)	Node numbers between which this circuit element lies. NODE (1, J) is the primary node, and NODE (2, J) the secondary node.
NPEC (6,30)	SPEC array in fixed format, used to hold the circuit element's number in NPEC (1, I).
NPOW (2, 1)	Array-holding resistor codes for converting voltage to current. NPOW (1, I) holds the circuit element number.
NPW	Read-in code for a value of the resistor to be used to convert voltage sources to current. This is stored in the NPOW array.
NR	Number of errors found on a Circuit Element Card.
NRDC	0 or 1, depending on whether we have read in the imaginary values of a special distribution for an A or Z circuit element.
NRDS	0 or 1, depending on whether we have read in the imaginary values of special values for an A or Z circuit element.
NSENSE (201)	Array containing ones and zeros, depending on whether the J circuit element is to be varied in a sensitivity solution.
NSPEC (201)	Array which tells the program if this circuit element has special values in addition to those in columns 41-80 of the Circuit Element Card.
NTYPE (201)	Circuit element type, either R, C, L. A, Z, E, V, D, I, B, H, F, or G.
NUM (2)	Circuit element reference number for a B, H, F or G element. Number NUM (2) is used for equation input. If NUM (2) = 0, no follow-on cards are needed.
R	Resistor circuit element.

CHECK (Continued)

SPEC (6, 30)	NPEC array holding special values for some of the circuit elements.
TRANS (2, 20)	Array which holds the circuit element numbers of B, H, F or G elements and the circuit element numbers of their reference elements.
V	DC voltage circuit element.
XHIGH (201)	High value of the circuit element for sensitivity analysis, or a special value.
XLOW (201)	Low value of the circuit element for sensitivity analysis, or a special value.
XMCHI (201)	Standard deviation of the Monte Carlo distribution (if a normal or lognormal distribution is required), or a special value.
XMCLO (201)	Mean of the Monte Carlo distribution (if a normal or lognormal distribution is required), or a special value.
XNOM (201)	Nominal value of circuit elements.
Y	AC current circuit element.
Z	Impedance-type circuit element.

CHECK calls no subroutines.

COMPUT (ITYPE, INODE, JNODE, IT, B, TRANS, V, E, D, Y, ND1, NODE, OUT, NDC, NODMX1, NEQU, NDEBUG, NER, R, H)

COMPUT is called by the main program after control has returned from subroutine SOLUT. COMPUT, called once for each circuit solution required by SNAP II, constructs the admittance matrix from the equivalent circuit. The simultaneous equations are node equations, and the right-hand sides of these equations are current. The first equation formed represents the current going into and out of the first node of the equivalent circuit. The right-hand side of the second equation, for example, would equal only active-current sources attached to node 2 of the equivalent circuit. (Realize that node 2 of the equivalent circuit might equal node 6, say, of the read-in circuit, in which case NEQU (7) = 2.)

The coefficients of the second equation are formed as follows: The coefficient for the first position (a_{21}) is the negative sum of the admittances of all circuit elements which lie between nodes one and two. The coefficient for the second position (a_{22}) is the sum of all the admittances of circuit elements attached to node two. The coefficient for the third position is minus the sum of all the admittances of circuit elements which lie between nodes two and three. Thus COMPUT must calculate the admittance from the impedance and put the sums in the correct element of the matrix A. Since A is a complex, double precision matrix, and the complex indexing and computation must be done by hand in the program rather than by the computer.

The dependent sources (B, H, G and F types) require more programming. If a dependent source lies between nodes MB and KB, and if its value depends on the current through a circuit element lying between nodes JB and IB, then in equation MB and KB the terms for nodes IB and JB will have a sum added to and subtracted from them, respectively. This sum will depend on the type of dependent source with which we are dealing.

COMPUT calls subroutine SOLVE to compute the node voltages. Before returning to the main program, COMPUT puts these node voltages in the array OUTPUT so they are in the right element for the read in node names.

A	Admittance-type circuit element.
B	Voltage-dependent current source.
BVAL	Value of a dependent power source.
D	DC current circuit element.
DIV	Double-precision reciprocal of the magnitude of a complex impedance.
E	AC voltage circuit element.
H	Voltage-dependent voltage source circuit element.
IB	One of the nodes between which the reference circuit element of a dependent power source lies. JB is the other node.
ICOMP (400)	Circuit element numbers in the equivalent circuit.

COMPUT (Continued)

II	2*I, indexes the imaginary part of an array.
IM	Temporary storage for the imaginary part of a sum in double precision.
INODE (400)	One of the nodes between which a circuit element in the equivalent circuit lies.
IR	II-1, indexes the real part of an array.
IT	Index of the TRANS array which holds the circuit element number of a B, H, F or G element and its associated reference element number.
ITYPE (400)	The circuit element types in the equivalent circuit. Either R, L, C, V, E, D, I, A, Z, B, H, F, or G.
JB	One of the nodes between which the reference circuit element of a dependent power source lies. IB is the other node.
JNODE (2, 51)	An array which serves as a table of contents for the array describing the equivalent circuit. See appendix on Network Storage.
K	Temporary value of a node number.
KB	One of the nodes between which a dependent power source lies. MB is the other node.
KBR	Index addressing real part of the KB element.
KI	Imaginary index of the Kth element.
KR	Real index of the Kth element.
MB	One of the nodes between which a dependent power source lies. KB is the other node.
MBR	Index addressing the real part of the MB element.
N	Reference circuit element number for a dependent power source.
NB	The beginning range of a DO loop which searches through one node's entries in the equivalent circuit's arrays.
NC	0 or 1 depending on whether NDC = 1 or 0.
NCOMP (201)	A place saver in COMPUT.
ND	Number of nodes in the equivalent circuit.

COMPUT (*Continued*)

ND1	Number of nodes in the equivalent circuit plus one.
ND2	2*ND
NDC	0 for DC circuit, 1 for AC circuit.
NDEBUG	1 or 0 depending, respectively, on whether or not we want intermediate debugging prints.
NEQU (1)	An array of node name equivalences between the read in circuit (contained in the index + 1 of NEQU) and the node number of the equivalent circuit. Thus NEQU (5) = 3 means that node 4 of the read in circuit is numbered node 3 of the equivalent circuit.
NER	0 or 1 on return from subroutine SOLVE, depending on whether or not the matrix was solvable.
NF	The end of a range of a DO loop. See NB.
NODE (2, 1)	The node numbers between which the circuit element lies. NODE (1, J) is the primary node, NODE (2, J) the secondary node.
NODMX1	The number of nodes plus one in the read-in circuit.
OUT (2, 1)	The output array of node voltages. OUT (1, 7) = the node voltage, real part, off of node 6. This is called OUTPUT in other subroutines.
R	Resistor-type circuit element.
RE	Temporary storage for the real part of a sum in double precision.
RVAL	The value of the reference circuit element of a dependent power source.
TRANS (2, 20)	The array which holds the circuit element numbers of B, H, F, or G elements and the circuit element numbers of their reference elements.
V	DC voltage circuit element.
VALUE (2,400)	The impedance of the circuit element in the equivalent circuit. IVALUE (1, I) = the real part, IVALUE (2, I) = the imaginary part.
Y	AC current circuit element.

COMPUT calls subroutine SOLVE.

CONECT (INODE, JNODE, ND, NDR, NO, NS, N)

CONECT is called once by ACEQ and DCEQ when a new equivalent circuit is being formed. CONECT tests the connectivity of the circuit network formed in ACEQ or DCEQ. The argument N controls whether this is the first time CONECT has been entered for this equivalent circuit, in which case we find the first node in the JNODE array and break the circuit into two parts: the portion of the circuit connected to the first node by a series of links (stored in NSAVE), and those not connected to the first node (stored in NOTCON).

If this is not the first time CONECT has been entered for this circuit, then N will equal the node we are to begin with. CONECT was constructed this way because if the calling program finds that the NSAVE array doesn't have a power source, then the part of the circuit with a power source is in NOTCON, which may consist of several unconnected networks. Thus N is a node in the NOTCON array constructed in the previous call to CONECT.

The method used in CONECT is a simple algorithm which traces up and down the tree formed by the network, starting with the first node. The nodes connected to the first node are located, the first node and its connected nodes are put in NSAVE, and the particular node being traced is kept track of. We then pick a new node from those in NSAVE, check the nodes connected to it to find if they are already in NSAVE. If not, they are added. We continue until we have tested all the nodes in NSAVE and don't find any new ones.

NOTCON is then constructed by filling it with the nodes in the equivalent circuit which were not in NSAVE. If there were no such nodes, NOTCON is empty on return.

DUM (601)	Place holder in the COMMON statement.
INODE (1)	Node in the equivalent circuit. See appendix on Network Storage.
JNODE (2, 1)	Table of contents for the equivalent circuit arrays.
N	Number of the node to be used as the first node.
NB	Starting value for a DO loop searching through the links attached to one node.
ND	Maximum index of JNODE in the equivalent circuit.
ND1	ND + 1.
NDR	Actual number of nodes in the equivalent circuit.
NF	Final value for a DO loop. See NB.
NFIRST	First or source-attached node in the equivalent circuit.
NO	Index of NOTCON. NO = 0 return to the calling program if the circuit is connected.
NOTCON (50)	Array of nodes not connected to the nodes of NSAVE. They are not necessarily connected to each other.

CONECT (Continued)

NOW	Node being considered. The links attached to this node are examined to determine if other nodes also common with these links should be added to NSAVE.
NS	Index of NSAVE. NS = NDR if the circuit is connected.
NS1	Used to find the next node.
NSAVE (50)	The array of nodes connected to each other.
VAL (1202)	Place holder in the COMMON statement.

CONECT calls no subroutines.

CURENT (I, ITYPE, INODE, JNODE, HEQU, ND, R, B, KS, DIAG, IP, NPOW, XNOM, JCOMP, V, E, H, G)

CURENT is called by subroutine SOLUT once for each voltage source (whether a V, E, H, or G type circuit element), for each solution required. If this is a new equivalent circuit, CURENT finds the voltage source's entry in the NPOW array and uses the number it finds there as a code for the resistor to be put in parallel with the voltage source in the equivalent circuit. If this is not the first time the equivalent circuit has been used in this run, then the resistor is already part of the equivalent circuit and we only need to look up its value in the NSAVE array, which was constructed during the first solution of this equivalent circuit. CURENT also sets an entry in the NSCR array to flag SOLUT that this voltage source has been converted to current so that when SOLUT sees the reflection of this circuit element, it will not call CURENT again.

B	Voltage-dependent current source. Not used in this version of CURENT.
DIAG	A dummy variable previously used for diagnostics.
E	AC voltage source.
G	Current-dependent voltage source.
H	Voltage-dependent voltage source.
I	Index of the voltage source in the equivalent circuit; there are two for each circuit element, one of which is I.
ICI	The voltage-source circuit element number.
ICOMP (400)	The circuit element numbers referencing the read-in circuit.
INODE (1)	One of the nodes between which a circuit element in the equivalent circuit lies.
IP	The number of entries in the NPOW array and equal to the number of voltage sources in the read-in circuit.
ITI	Type of voltage-source circuit element, either V, E, F, or G.
ITYPE (1)	Circuit element type in the equivalent circuit, either a R, L, C, H, Z, E, V, I, D, B, H, F, or G.
IV	Value of the resistor to be put in parallel with the voltage source when we are computing it from the NPOW array.
IVALUE (2, 1)	Value of a circuit element for this solution of the equivalent circuit.
JC	Two times the number of circuit elements in the equivalent circuit.
JCOMP	Number of circuit elements in the read-in circuit.

CURENT (Continued)

JNODE (2, 1)	An array which directs the program in the INODE, ITYPE, IVALUE, ICOMP arrays. See appendix on Network Storage.
JSC	The number of circuit elements in the read-in circuit plus KS, the number of resistors added.
K1	One of the nodes (K2 is the other) between which the voltage source lies. (This node is in the equivalent circuit and may not have the same number as the node in the read-in circuit.)
K2	One of the nodes (K1 is the other) between which the voltage source lies in the equivalent circuit.
K22	An index in the JNODE array for use in inserting the resistors into the equivalent circuit storage.
KCOMP	The read-in index or circuit element number of the voltage source we are converting to current.
KS	The number of entries in the NSAVE array. If this is the first solution for this equivalent circuit, KS = 0.
KV1	Index in the equivalent circuit arrays of the first use of the voltage-source circuit element we are converting.
KV2	Index in equivalent circuit arrays of the second use of the voltage-source circuit element we are converting.
NB	The beginning range of a DO loop used to search through one node's links for a particular circuit element.
NCOMP (201)	A dummy place holder for COMMON storage.
ND	Number of nodes in the equivalent circuit.
ND1	ND, the number of nodes in the equivalent circuit.
NEQU (1)	An array of node equivalences. When an equivalent circuit has been established, we renumber the nodes. The index of NEQU minus one is the old read-in circuit node number, while the value of NEQU (I) is the equivalent circuit renumbering of that node.
NF	The end of a DO loop used to search through all the links from a particular node.
NPOW (2, 1)	An array of voltage-source circuit element numbers (NPOW (1, I)) and the negative exponent of their associated resistors to be put in parallel with them.

CURENT (Continued)

NSAVE (2, 25)	An array of voltage-source circuit element numbers (NSAVE (1, I)) and the negative exponent power of their parallel resistors (NSAVE (2, I)). NSAVE is constructed when we first add the resistors to an equivalent circuit.
NSCR (402)	An array used to flag voltage sources in the equivalent circuit which have already been converted to current sources. This keeps SOLUT from calling CURENT a second time for the same circuit element in single solution.
R	Resistor.
V	DC voltage source.
VOLT	The real value of the voltage source for this solution. VOLT will change sign for the two entries in IVALUE to reflect the direction of current flow.
VOLT2	0 unless the value for the voltage source was supplied by EQUIN.
XNOM (1)	Nominal value of the circuit elements in the read-in circuit.
XR	The value of the resistor to be put in parallel with the voltage source to make it a current source. XR is computed from an entry in NPOW (2, I) or from NSAVE (2, I).

CURENT calls no subroutines.

CURR (N)

CURR computes branch current through a circuit element, N, whose circuit element number is CURR's only argument. CURR is only called by subroutine EQUOUT when current is a value needed by an equation.

CURR	The current we computed and are returning to subroutine EQUOUT.
ICOMP (400)	An array of the circuit element numbers of the elements in the equivalent circuit. The engineer may have requested branch current through a circuit element dropped on conversion to the equivalent circuit.
INODE (400)	A place saver in the COMMON statement.
IVALUE (2, 400)	A place saver in the COMMON statement.
N	The circuit element number of the element through which we must find the current.
NCOMP (201)	A place saver in the COMMON statement.
NCON (100)	A place saver in the COMMON statement.
NEQU (51)	An argument needed by subroutine BRANCH.
NODE (2, 201)	An argument needed by subroutine BRANCH.
NSCR (2, 201)	A place saver in the COMMON statement.
OU (2)	The result of calling BRANCH. The real part of the current is in OU (1), the imaginary part is in OU (2).
OUTPUT (2, 51)	An argument needed by subroutine BRANCH.

CURR calls subroutine BRANCH.

DCEQ (NODE, NTYPE, ITYPE, INODE, ND, NODMAX, E, Y, L, DIAG, C, V, D, NEQU, JNODE)

DCEQ is called by the main program whenever we have a DC circuit solution request and this is either the first solution request after reading in the component cards, or the previous solution request was for an AC circuit. DCEQ converts the read-in circuit to a DC equivalent circuit. This is accomplished by shorting AC voltage sources and inductors, and opening AC current sources and capacitors. In the first case, we lose a node and those components in parallel with the shorted component. In the case of capacitors, we just "disconnect" the component.

Since opening circuits can cause the network to be unconnected, we call subroutine CONECT to test the connectivity of the network. If the network is now in two or more pieces, some without a power source, the pieces without power are dropped from the equivalent circuit. DCEQ may have to call CONECT several times to eliminate all such pieces. When all such unpowered, unconnected sections of the network are eliminated, we construct the equivalent circuit from the scratch array, NSCR, in which we have been flagging the nodes we wish to eliminate. Forming the equivalent circuit consists of going through the scratch array and finding which nodes are connected to each other. DCEQ also, at this time in the program, eliminates any node numbers which do not have links coming from them. This compresses the network and makes the "table of contents" array, NEQU, necessary.

Finally, the program tests that at least one node is designated "0" for ground--that is, "0" was not an eliminated node--and then returns to the main program.

C	Capacitor-type circuit element.
D	DC current circuit element.
DIAG	0 usually. If subroutine CONECT finds the circuit without power, DIAG = 1.
E	AC voltage circuit element.
I1	I + 1, necessary when examining arrays concerning nodes, since ground is zero.
ICOMP (400)	An array of circuit element numbers which serve as an index between the read-in circuit and the equivalent circuit.
INODE (1)	One of the nodes between which lies a circuit element in the equivalent circuit.
IOUT	0 if no danglers (links without attachment to other links on one end) are found after all the appropriate links were dropped; 1 if danglers are found.
IPOW	Controls whether or not the part of the network in NSAVE is powered.
ITYPE (1)	Type of circuit elements in the equivalent circuit, either R, L, C, V, E, D, I, A, Z, B, H, F, or G.

DCEQ (Continued)

IVALUE (2, 400)	A place saver in the COMMON array.
JC	Twice the number of circuit elements in the equivalent circuit.
JNODE (2, 1)	An array which serves as a table of contents for the array describing the equivalent circuit. See appendix on Network Storage.
K	Temporary storage for a particular value of ICOMP. This is necessary since the entries of ICOMP are themselves used as indices.
K1	Temporary storage for an element of the JNODE array.
K2	Temporary storage for an element of the JNODE array.
M	Temporary storage of a node in the read-in circuit.
N	Temporary storage of a node in the read-in circuit.
NB	Beginning range of a DO loop which searches through one node's entries in the equivalent circuit's arrays.
NCOMP (201)	An array of circuit element numbers in read-in circuit order. In fact, NCOMP (N) = N after processing by EXCH.
ND	Maximum node in the equivalent circuit.
ND1	The number of nodes in the equivalent circuit, plus one (to account for ground being zero).
NDR	A counter of the number of nodes with links in the equivalent circuit. Since some nodes may be skipped at this point in the program, NDR may be less than ND, the maximum node in the equivalent circuit.
NEQU (1)	An array of node name equivalences between the read-in circuit (contained in the index + 1 of NEQU) and the node number of the equivalent circuit. Thus NEQU (5) = 3 means that node 4 of the read-in circuit is numbered node 3 of the equivalent circuit.
NF	The end of a range of a DO loop. See NB.
NND	Temporary storage for ND + 1 when we are constructing the compressed equivalent circuit.
NODE (2, 1)	The node numbers this component lies between. NODE (1, J) is the primary node, NODE (2, J) the secondary node.
NODMAX	The maximum node in the read-in circuit.

DCEQ (Continued)

NODMX1	The maximum value of the nodes plus one in the read-in circuit.
NOTCON (50)	An array of nodes which, after return from subroutine CONECT, the program has found unconnected to the nodes in NSAVE.
NSAVE (50)	An array of connected nodes, as distinct from NOTCON, in the newly generated equivalent circuit.
NSCR (2, 201)	An array identical with the NODE array (of the read-in circuit) at the beginning of this subroutine. Gradually NCSR is transformed into an array containing the node linkages of the new equivalent circuit.
NTYPE (1)	The circuit element type, either R, C, L, A, Z, E, V, D, I, B, H, F, or G.
V	DC voltage circuit element.
Y	AC current circuit element.

DCEQ calls subroutine CONECT.

EQUIN (N, INPUT, DUM, VOLT, FREQ, ERR)

EQUIN is called by subroutine ASSIGN when a circuit element's value is expressed as a functional relationship rather than read in as a constant from the Circuit Element Card. The engineer prepares a GO TO card and the equations in FORTRAN IV. The integer N refers to the equation number and is also a statement number of the FORTRAN arithmetic statement prepared by the engineer. On return to ASSIGN, the impedance of the circuit element is in INPUT.

CARD (M)	Impedance of the Mth circuit element.
CURR (N)	Branch current through the Nth circuit element, used only for nonlinear solutions.
DUM	A space saver so that the value INPUT has real and imaginary values.
ERR	0 if an equation exists for this value of N; 1 otherwise.
FREQ	Frequency at which equivalent circuit is being solved.
INPUT	Impedance of the circuit element on return to ASSIGN.
N	The equation number to be used in computing this circuit element's value.
VOLT	The voltage off the Nth node, used for nonlinear solutions.

EQUIN calls no subroutines.

EQUOUT (N, VOLT, OUTPUT, FREQ, ER)

EQUOUT is called by subroutines POLAR, SENSPR, SENSP1, PLOTF, and STAT. It computes functional outputs from node voltages, frequency, circuit element values, and branch currents. The engineer prepares the equations, each of which must have a unique sequential statement number.

CARD (M)	Impedance of circuit element M.
CURR (M)	Branch current through circuit element M.
ER	0 if there is an equation whose number is in EQUOUT. If ER = 1, this signals subroutine POLAR that there is no equation with that number in it.
FREQ	The frequency at which the node voltage in VOLT was obtained.
N	The unique equation number, which is identical with one of the statement numbers in EQUOUT.
OUTPUT	Result of the calculation performed in EQUOUT. OUTPUT (1) is the real part, OUTPUT (2) is the imaginary part.
VOLT (M)	Node voltage of node M.

EQUOUT calls no subroutines.

EXCH (I, J, NTYPE, NAME, NODE, NSENSE, NDIST, XNOM, XHIGH, XLOW, XMCLO, XMCHI, NSPEC, NMC, N2, IS, IC, IAD, IE, IT, B, A, Z, NPEC, NARLO, TRANS, NDMIT, NEQUAT, IP, NPOW, V, E, H, G, F)

EXCH is called by the main program once for each circuit element in the data deck. The engineer is not required to number (in columns 1-3) all circuit elements, but only those referenced by others. However, to save computer storage and simplify programming, it is necessary to number all the circuit elements after they have been read in. The main program does this, assigning numbers to the circuit elements not numbered. Then subroutine EXCH puts this new circuit element number in the NPOW, TRANS, NEQUAT, ADMIT, and SPEC arrays where appropriate, and puts the element's entries in the other arrays in the position it should have with its new number. EXCH is not called again once the circuit has been rearranged.

A	Admittance-type circuit element which requires an entry in the NDMIT array for imaginary values.
B	Voltage-dependent current source. This requires a reference circuit element number in the TRANS array.
E	AC voltage source.
F	Current-dependent current source.
G	Current-dependent voltage source.
H	Voltage-dependent voltage source.
I	Position in the circuit element arrays that this element currently occupies.
IAD	Number of circuit elements with entries in the NDMIT or ADMIT array.
IC	Number of circuit elements with entries in the NARLO or CARLO array.
IE	Number of circuit elements whose value depends on an equation in subroutine EQUIN. Thus there are IE entries in NEQUAT.
IP	Number of voltage circuit elements (i. e., the number of entries in NPOW).
IS	Number of circuit elements with entries in the SPEC or NPEC array.
IT	Number of circuit elements with entries in the TRANS array.
J	Circuit element number and position in the element arrays that a circuit element will occupy when EXCH returns to the main program.

EXCH (Continued)

N2	Number over 200 used as the circuit element number until EXCH was called for this circuit element.
NAME (1)	Up to 4 alphanumeric characters labeling the circuit element.
NARLO (41, 1)	Array of special distributions to be used when varying this circuit element in a Monte Carlo solution. NARLO (1, I) contains the circuit element number.
NCOMP (201)	Array of circuit element numbers. When EXCH has been called for all the circuit elements, NCOMP (I) = I.
NDIST (1)	0 if the circuit element is going to take on its nominal value in a Monte Carlo solution. If NDIST (1) = 4, there are special distribution entries in array NARLO.
NDMIT (6, 1)	Array of imaginary parts for any A or Z circuit element. NDMIT (1, I) contains the circuit element number.
NEQUAT (40, 1)	Array of circuit element numbers (in NEQUAT (1, I)) and equation numbers for subroutine EQUIN (in NEQUAT (2, I)) for those circuit elements whose value depends on functional relationships. NEQUAT (3, I) is zero, 1, or 2 depending on whether 0, 1 or 2 follow-on cards followed the Circuit Element Card.
NMC (1)	Number of points in a circuit element's special distribution (if NDIST (I) = 4).
NODE (2, 1)	Two nodes between which the circuit element lies.
NPEC (6, 1)	Array of special values for the circuit element. NPEC (1, I) = the circuit element's number, and NPEC (2, I) through NPEC (6, I) the special values.
NPOW (2, 1)	Array of resistor negative exponents to use in converting voltage (V, E, G, H circuit element types) to current. NPOW (1, I) contains the circuit element number.
NSENSE (1)	0 if the circuit element is not going to be varied in a sensitivity output; otherwise, NSENSE = 1.
NSPEC (1)	An array of special values which the circuit element can use in a special solution.
NT	Temporary storage used by EXCH.
NTYPE (1)	The letter A, Z, R, L, C, V, E, I, D, B, H, G, or F describing the type of circuit element.
TRANS (2, 20)	Array of reference circuit element numbers for use with B, H, G, and F types. TRANS (1, I) contains the B, H, G, or F circuit element number.

EXCH (Continued)

V	DC voltage source.
XHIGH (1)	Sensitivity high value of the circuit element (or a special value).
XLOW (1)	Sensitivity low value of the circuit element (or a special value).
XMCHI (1)	Monte Carlo standard deviation or a special value.
XMCLO (1)	Monte Carlo distribution mean value, or a special value.
XN	Temporary storage used by EXCH.
XNOM (1)	Nominal value of circuit element.
Z	Impedance-type circuit element requiring an entry in the NDMIT array.

EXCH calls no subroutines.

HIST (NXT, RCT, LABEL, LEQU, SIG)

HIST is called by subroutine STAT when all the Monte Carlo variations have been performed by the program and the results are ready to be plotted in a histogram. HIST is called once for each output.

The array of points to be plotted must be returned to STAT without changes, since STAT prints out a listing of their range after calling HIST. Therefore HIST stores the array of ordinates in a temporary array which it then scales so the histogram will fit on a page. This scaling is accomplished by dividing the ordinates by 2 until their maximum is less than 50. The abscissa is divided up two different ways, depending on the number of samples used in the Monte Carlo analysis. If NORV, the number of samples read in after the type card, is less than 300, then the interval is broken up into 12 intervals with a bar at 1/2-sigma intervals about the mean. If NORV is greater than 300, the histogram is broken up into 24 intervals with a bar at 1/4-sigma intervals. The labeling is at 1/2-signal intervals, however.

HIST starts at the top of the page (the title has already been printed in STAT); puts blanks in the printing array, LPRNT; and then uses one of two parts of coding, depending on whether we are printing 12 bars or 24. In each case, the ordinate array is tested to see if there are any entries that would be full enough to reach the top of the page. If so, it is necessary to enter asterisks or other symbols into that bar representing that range of ordinates. When we have tested all the ordinates for the top of the page printout, we print the line of print, go to the end of the print loop, and start over (first blanking the line) on the next to top line. This procedure is continued until the histogram has been printed.

Finally, the bottom line is printed after testing to see if there are 12 bars or 24. The labeling of this bottom line is independent of the number of bars printed and consists of letters ±SG1, ±SG2, etc., and the numeric value for these points.

HIST then returns control to STAT.

AST1	"*", used to form the left hand margin.
COLMID	Four dots used for the inside of a double bar when NXT = 12.
COL8L	3 blanks and a 1, thus " 1" used to form the bar when NXT = 24 and the preceding KCT was zero.
COL8R	The characters "...1" used to form the bar when NXT = 24 and the preceding KCT was not 0.
F13 (19)	Bottom line of the histogram if NXT = 12.
F25 (19)	Bottom line of the histogram if NXT = 24.
J	Index which keeps track of the LPRNT index. Every word of LPRNT controls four characters across the line being printed so if J = 5, we are looking at the 17-20 characters of a line of printing.
K	A reverse index. The printing DO loop goes from 1 to 50, but we start at the top of the page, so we must test the KCT array to see if it has elements equal to 50 (or to K). As the DO loop proceeds and IP increases, K decreases.

HIST (Continued)

K2	Scaling number used to keep track of how many times we had to divide KCT elements by 2 before getting the maximum KCT element below 50.
KCT (25)	RCT array scaled so it will fit in the 50 lines of a computer page.
L1	Characters "*1" used for the left hand margin if a bar must be printed there.
LABEL (25)	Array used for labeling the abscissa axis. Label contains blanks and words like "+SG1", "-SG1", etc.
LB4	Four blank characters, used to initialize.
LEQU (2)	The label for this output. Subroutine STAT transmits the LEQU (2, I) and LEQU (3, I) entries of main to HIST so they look like LEQU (1) and LEQU (2).
LPRNT (28)	A print array containing the characters to be printed for one line of the histogram. LPRNT is formed by testing KCT to see if the KCT entries would cause any histogram printout at that position of the histogram.
MAX	Maximum KCT element.
MIN4	Characters "---1" used to form the top of the bar when NXT = 24.
MINMID	Characters "----" used to form the first part of the top of the bar when NXT = 12. The right half of the top of the bar is formed using MIN4.
NEW	0 if the adjacent bar was not printed, 1 if we don't need to print a beginning bar.
NXT	Number of entries in RCT, the ordinate array. NXT is 12 or 24, depending on whether NORV (in the main program) was less than or greater than 300, respectively.
RCT (1)	Array of integers used to obtain the ordinate values of the histogram. In STAT, when an output is within a certain range, we add one to that range's RCT entry.
SIG (7)	Numeric values of the $\pm 3\sigma$ points, SIG (7) and SIG (1); $\pm 2\sigma$ points, SIG (6) and SIG (2); $\pm 1\sigma$ points, SIG (5) and SIG (3); and the mean (SIG 4) used for labeling.
YL	Ordinate axis labels giving the value of KCT. YL is blank for four lines, and then prints the value of the fifth entry of KCT on every fifth line.

INOUT (JCOMP, IE, NEQUAT, NTYPE, NAME, NODE, A, Z, NSENSE, NDIST, XNOM, XLOW, XHIGH, XMCLO, XMCHI, NSEC, NMC, IAD, ADMIT, IS, SPEC, CARLO, IT, TRANS, IC)

INPUT is called by the main program once for each original circuit after EXCH has been called for all circuit elements. INOUT prints each circuit element and the values and information which appeared on its Circuit Element Card. INOUT checks to see if the circuit element's value is determined by an equation rather than numeric input, and checks to see if the element is an admittance or impedance type. (If so, it must print real and imaginary parts for it.) Finally, INOUT prints special arrays formed in CHECK for special Monte Carlo distributions.

A	Admittance-type circuit element.
ADMIT (6, 1)	Array used to hold the imaginary parts of A and Z circuit elements. ADMIT (1, I) is called NDMIT and holds the circuit element's number.
CARLO (21, 30)	Same as NARLO, but with a floating point name.
IAD	Index of the ADMIT array, used for imaginary parts of A and Z circuit elements.
IC	Index of the CARLO (or NARLO array).
IE	Index of NEQUAT array, which holds the circuit element number and its equation number.
IS	Index used by the NPEC and SPEC array.
IT	Index of the TRANS array which holds the circuit element number of a B, H, F, or G element and its associated reference element's number.
JCOMP	The number of circuit elements in the read-in circuit.
NAME	Up to 4 alphanumeric characters used to describe the circuit element, besides its type.
NCOMP (201)	The unique circuit element number assigned by the engineer or by CHECK.
NDIST (201)	Array of either 0, 1, 2, 3 or 4 giving the codes for the Monte Carlo distributions of the circuit elements.
NDMIT (6, 50)	The fixed point array of ADMIT.
NEQUAT (40, 30)	An array of circuit element numbers (in NEQUAT (1, I)) and their associated equation numbers (in NEQUAT (2, I)) if their value must be computed. NEQUAT (3, I) to NEQUAT (39, I) is used for follow-on cards.

INOUT (Continued)

NMC	An array which tells the program, if this circuit element has special distribution for Monte Carlo analysis, how many points are in the distribution.
NODE	The node numbers between which this circuit element lies. NODE (1, J) is the primary node, NODE (2, J) is the secondary node.
NSENSE (201)	An array containing ones and zeros depending on whether the J circuit element is to be varied in a sensitivity solution.
NSPEC (201)	An array which tells the program if this circuit element has special values in addition to those in columns 41-80 of the Circuit Element Card.
NTYPE (201)	The circuit element type, either R, C, L, A, Z, E, V, D, I, B, H, F, or G.
SPEC (6, 30)	The NPEC array holding special values for some of the circuit elements.
TRANS (2, 20)	The array which holds the circuit element numbers of B, H, F, or G elements, and the circuit element numbers of their reference elements.
XHIGH (201)	High value of the circuit element for sensitivity analysis, or a special value.
XLOW (201)	Low value of the circuit element for sensitivity analysis, or a special value.
XMCHI (201)	Standard deviation of the Monte Carlo distribution (if a normal or lognormal distribution is required), or a special value.
XMCLO (201)	Mean of the Monte Carlo distribution (if a normal or lognormal distribution is required), or a special value.
XNOM (201)	Nominal values of the circuit elements.
Z	Impedance-type circuit element.

INOUT calls no subroutines.

NOLIN (NODNAM, GUESS, TOL, NEQU, OUTPUT, ND, INDIC, NDEBUG, NOLIN, NAC, GUESLO, GUESHI)

NOLIN is called by the main program once at the beginning of a solution to initialize the first guesses into the output array, and is subsequently called each time the main program has a new solution. NOLIN is only called when a nonlinear set of equations is input through subroutine EQUIN.

On initialization, if the engineer did not supply upper and lower bounds, they are set to ±50 percent of the first approximation supplied by the engineer. If no tolerance limit was set, the tolerance is set to 0. 2 percent of the first approximation. If the tolerance provided was smaller than 0. 01 percent of the first approximation, it is set at 0. 2 percent of the first approximation.

After the first solution, using the first approximation, has been found, the program finds the "area" that the solution falls into and decides on a delta to vary the next approximation. The areas are defined as follows:

Area 1: The difference between the approximation used and the output obtained from this approximation is less than ±20 times the tolerance.

Area 2: Between area 1 and the high or low limits.

Area 3: Above or below the high or low limits.

The logic used to obtain a solution differs considerably for these three areas, and depends on the preceding delta, the differences, the preceding approximations, and the resulting outputs.

When the square of the sum of the differences (between the output obtained and the approximation used) of the nonlinear outputs is less than the square of the sum of the tolerances, we have a solution and return to the main program with INIC set to 2 to indicate this.

If we have iterated more than 20 times the number of nonlinear outputs, we return to the main program with the INDIC set to 3 to indicate that no solution was obtained and that we should stop iterating.

DEL	Temporary storage for DELTA (I).
DELT (9)	Increment added to TRY (1, I, 2) to obtain the next approximation. DELT (J) is the delta used for the node in NODNAM (J).
DIF (9, 2)	Array containing the differences between the output obtained and the approximation used. DIF (3, 1) is this difference for the node found in NODNAM (3). The difference between the approximation used to obtain the present output and the previous approximation (in TRY (1, 3, 3)) is found in DIF (3, 2). On return to the main program, when we have decided on a new approximation, DIF (3, 1) contains the difference between approximations, not the approximations and outputs themselves.
DSAVE (9)	Not used in this version of NOLIN.

NOLIN (Continued)

GSAVE (2, 9)	Previous output voltage obtained for a particular node. GSAVE (1, 3) is the previous output of the node in NODNAM (3).
GUESHI (2, 1)	High limits of the nodes in NODNAM. See GUESLO.
GUESLO (2, 1)	Low limits of the nodes in NODNAM. The first unknown node is in GUESLO (1, 1). To find which node it is in the equivalent circuit, we must look in NODNAM (1).
GUESS (2, 1)	First approximations of the values of the nodes in NODNAM. Always supplied by the engineer.
INDIC	Indicator whose value flags the main program as to progress in NOLIN, INDIC = 0 means this is the first time we've entered NOLIN and initialization is necessary. INDIC = 1 means we are still trying to achieve a solution. INDIC = 2 means we have a solution, and INDIC = 3 means we have not achieved a solution and are giving up.
K	Node number of the node voltage we are presently approximating.
LIM	A counter which counts the number of iterations performed. When LIM is greater than LIMIT, INDIC is set to 3 and we return to main.
LIMIT	Number of iterations possible before setting INDIC = 3 and giving up on a solution. At the present time, LIMIT is set to 20 times the number of nonlinearities.
NAC	0 if this is a DC circuit. If NAC = 1, this is an AC equivalent circuit, which can't be in NOLIN.
NC	Temporary storage for NCLOS (I).
NCLOS (9)	Tag which is 1, 2, or 3 (plus or minus) depending on what area the solution was from the previous iteration. Thus if NCLOS (2) = 1, it means the output from the node in NODNAM (2) is less than 20 times the tolerance from the approximation used to obtain this output (i.e., the voltage area is area 1).
ND	Number of nodes in the equivalent circuit.
NDEBUG	Read-in variable from the main program. NDEBUG = 0 when intermediate results are not wanted. NDEBUG = 1 when the initial values of NOLIN are not to be printed out.
NEQU	Array not needed in this version of NOLIN.
NODNAM (1)	Node numbers of the nodes whose output is dependent on other nodes, and which thus must be solved for iteratively in NOLIN.
NONLIN	Number of dependent nodes in this equivalent circuit.

NOLIN (Continued)

OUTPUT (2, 1)	Node voltages of the nodes in the read-in circuit. The output voltage of node 6 is found in OUTPUT (1, 7). Only real voltage is considered in NOLIN, so OUTPUT (2, 7) is not used.
SUMDIF (2)	Sum of the squares of the differences when they are still the differences between the outputs obtained and the approximations used to obtain them.
SUMTOL	Sum of the squares of the tolerance, used to compare with sumdif to see if we have a solution.
TOL (1)	Value for each unknown nonlinear node which tells us how close to a solution we must come before we can call it a solution.
TRY (2, 9, 3)	Array of approximations and outputs obtained. Three previous values of TRY have been kept, but the third one is no longer used in this version of NONLIN. While room for imaginary parts are available in TRY (2, I, J), it is not used. As an example, TRY (1, 3, 2) contains the value used to obtain this output voltage for the node contained in NODNAM (3). Thus it contains the previous approximation. TRY (1, 3, 1) will contain a new approximation on return to main, but until this is set, it will contain the voltage obtained from the preceding approximation.
X	+1 or -1. X, like Y, is used to find if a sign has changed.
Y	+1 or -1. Y, like X, is used to find if a sign has changed.

NOLIN calls no subroutines.

NORM (RV, SIG, XMEAN, VAL, IST)

NORM is called by the main program when a Monte Carlo solution is requested and when at least one circuit element in the equivalent circuit must be varied according to a normal distribution. NORM is called once for each such circuit element, except in the case of A and Z circuit element types, when it is called twice.

DIF	Difference between the closest value of XNORM and RV.
IST	0 the first time NORM is entered during a run. It causes NORM to compute the abscissas of the normal distribution.
NEG	0 if RV is less than 0.5; 1 when RV is greater than 0.5. When the normalized random variable is found, and if NEG = 1, VAL is rescaled.
NTOP	A counter which varies from 1 to 19 and causes NORM to vary the normalized random variable in increments of 0.1 when they would be greater than 8.5 or less than 1.5.
RV	The uniformly distributed random variable supplied to NORM and returned to main.
SIG	The standard deviation of the normal distribution desired. Supplied by main.
VAL	The normally distributed random variable computed by NORM and returned to main.
XM	Slope of the interpolated curve.
XMEAN	Mean of the normal distribution desired. Supplied by main.
XNORM	(2, 57) the array containing the normal distribution; the first entry is the abscissa, the second is the ordinate.

NORM calls no subroutines.

OUT (XMAG, XPHS, OUTPUT)

OUT is called by subroutines POLAR, SENSP1, PLOTF, STAT, and SENSP. Its only function is to convert rectangular coordinates to polar coordinates, which is necessary for AC output. The sign of the magnitude for DC output is preserved in OUT.

I	Always 1, and only necessary because of a previous version of OUT.
OUTPUT	The rectangular coordinates of the voltage output. The real part is in OUTPUT (1), the imaginary part is in OUTPUT (2).
RE	Temporary storage of OUTPUT (1, I).
XIM	Temporary storage of OUTPUT (2, I).
XMAG	The magnitude of the voltage output in the AC case; for DC circuits it is simply OUTPUT (1). This is computed in OUT.
XPHS	The phase of the voltage in the AC case. XPHS = 0 for DC equivalent circuits. XPHS must be adjusted by 180 degrees if it lies in quandrant II or III.

OUT calls no subroutines.

PLOTF (LF, NLF, LEQU, JS, OUTPUT, IWHICH, FREQ, MODE, LG, XMAX, XMIN, NYGRID, NFGRID, TTLPT)

PLOTF is called by the main program and used when a frequency plot solution is requested. If LF is less than NLF, we are still computing outputs at different frequencies and are not ready to plot. Instead, we compute the output desired from EQUOUT, convert it to magnitude and phase in OUT, and store it in the array STORE depending on the value of IWHICH.

When LF = NLF, we can plot the outputs saved in STORE. If MODE = 1, maximum and minimum values must be computed for each output and for the frequency range. These values are then adjusted so that 1) the grid lines fall on non-fractional points, and 2) the plot is centered on the space available, and does not run up to the top and bottom of the plot. If MODE = 2, we use the limits set by the engineer, but we do test to see that there are no more than 10 frequency grid lines, since labeling is clumsy and cramped if we do. PLOTF calls subroutines described in the North American Aviation, Inc., Engineer's Computing Manual for the General Dynamics SC-4020 Plotter.

A separate section of PLOTF is used for logarithmic plots.

PLOTF returns to the main program when either a group of outputs for one frequency have been stored (if LF is less than NLF), or when all the outputs have been plotted (when LF equals NLF).

B	A working variable used by PLOTF to obtain the correct maximum and minimum values of the output when MODE = 1.
BEND	The final output value plus a scaling factor to assure centering and nonfractional grid lines. BEND = XMAX if MODE = 2.
BSTART	Starting value of the output axis. BSTART = XMIN when MODE = 2, otherwise it is calculated.
DEL	The interval between grid lines along the output axis. This is computed in both the MODE 1 and MODE 2 cases.
DX	The floating-point form of the increment used between grid lines along the frequency axis.
ER	Error flag resulting from a bad equation in EQUOUT. Not tested in PLOTF.
F	The range of frequencies used.
FREQ (100)	The frequencies at which the solutions have been obtained. (Comes from the main program.)
I	A variable which controls the number of labeled frequency grid lines. When I = 1, every grid line along the frequency scale will be labeled. If I = 10, every 10th grid line will be labeled.

PLOTF (Continued)

IWHICH	1, 2, or 3, coming from the main program. IWHICH = 1 indicates that all outputs are plotted in magnitude; IWHICH = 2 indicates that all outputs are plotted in phase; and IWHICH = 3 indicates that both magnitude and phase are to be plotted.
J2	A variable which controls the number of labeled output grid lines. It acts as I does, but on the output axis.
JS	The number of outputs requested. If IWHICH = 1 or 2, there will be JS plots produced. If IWHICH = 3, there will be 2*JS plots. Supplied by the main program.
K	The value of J from 1 to JS (the number of outputs desired) if IWHICH = 1 or 2. If IWHICH = 3, K will also take on the values J + 5 from 1 to JS.
LEQ	Temporary storage for LEQU (1, I), which contains the equation number of the output desired (defined in PLOTF).
LEQU (3, 1)	Output equation number (LEQU (1, I)) and title (LEQU (2, I) and LEQU (3, I)) for each output. Supplied by the main program.
LF	The frequency index, supplied by the main program.
LG	0 or 1. If LG = 0, the frequency axis will be laid out logarithmically. If LG = 1, the frequency axis is linear. LG comes from the main program.
M	A variable which causes output grid lines to be retraced for emphasis. PLOTF sets them to the output values of J2.
MODE	1 or 2, comes from the main program. If MODE = 1, PLOTF computes the output maximum and minimum and number of grid lines. If MODE = 2, these are supplied by the main program.
N	A variable which causes the frequency grid lines to be retraced for emphasis. PLOTF sets them to the value of I.
NB	A temporary storage variable used when B/DEL must be truncated to obtain a nonfractional grid interval.
NF	Half the number of frequencies. It is used to compute the grid size of the frequency axis when MODE = 1.
NFG	The number of frequency grid lines adjusted to a number under 11 if too many grid lines were specified.
NFGRID	The number of grid lines along the frequency axis. This is computed in PLOTF if MODE = 1; it is supplied by the main program if MODE = 2, but may be corrected by PLOTF if it is too large. It is independent of output.

PLOTF (Continued)

NINT	The number of intervals on the output axis. NINT is used when MODE = 1.
NLE	The last frequency index, supplied by the main program. If LF = NLF, we store the last output and begin plotting STORE.
NX	A variable used by NXV function to connect the points of the plot.
NX2	A variable similar to NX.
NY	A variable used by NYV function to connect the points of the plot.
NY2	A variable similar to NY.
NYGRID (1)	The number of grid lines along the output axis. This is computed in PLOTF if MODE = 1; otherwise it is supplied by the main program.
OU	Computed output resulting from calling EQUOUT. OU (1) contains the real part, OU (2) = the imaginary part of the output.
OUTPUT (2, 50)	The node voltage outputs (Re = OUTPUT (1, I), Im = OUTPUT (2, I)) resulting from this frequency's solution of the circuit. The real part of the voltage at node 4 is found in OUTPUT (1, 5). OUTPUT comes from the main program.
RANGE	The difference between the maximum and minimum values for a particular output. When MODE = 1, RANGE must be computed to begin the computation of NYGRID.
STORE (100, 10)	Array defined in PLOTF and used to store the computed outputs which will be plotted when LF = NLF. For example, the second output for the fourth frequency is stored in STORE (4, 2).
TENPC	A factor added and subtracted from the computed maximum and minimum output for a MODE 1 plot. TENPC centers the plot on the space assigned to it.
TTLPT	A title supplied by the main program for each plot if MODE = 2.
XF	The floating point form of NF when NF has been adjusted to the number we think it should be (used when MODE = 1).
XMAG	Magnitude of the output OU.

PLOTF (Continued)

XMAX (10)	The maximum output for a mode 2 plot. If IWHICH = 3, XMAX (6) through XMAX (10) contains the phase maximums of up to 5 outputs. XMAX is computed by PLOTF if MODE = 1; otherwise it comes from the main program.
XMIN (10)	The minimum output for a mode 2 plot. If IWHICH = 3, XMIN (6) through XMIN (10) contains the phase maximums of up to 5 outputs. XMIN is computed by PLOTF if MODE = 1; otherwise it comes from the main program.
XNINT	The floating-point form of NINT.
XPHS	Phase of the output OU.
YD	The number of frequency grid lines when MODE = 2. If YD exceeds 10, the labeling will not be good, so in this case the number of grid lines is decreased.

PLOTF calls subroutines EQUOUT, OUT, CAMRAV, GRIDIV, PRINTV, APRNTV, APLOTV, LINEV, SMXYV, ENDPLT.

POLAR (NC, AC, FREQ, LF, NODMAX, NOUT, NEQU, OUTPUT, JS, LEQU, OU, NCUR, NODE, JCOMP, NTYPE, NAME, INODE, B, H, G, F, E, V, D, Y)

POLAR is called by the main program when a nominal or special solution has been requested and when the main program has solved the impedance matrix and obtained node voltages. POLAR computes the node voltages, branch currents, and special equation outputs requested by the engineer. If node voltages for nodes dropped on conversion to the equivalent circuit were requested, POLAR prints a list of these nodes. In the same manner, if branch currents through circuit elements that were dropped on conversion to the equivalent circuit or dependent or independent sources are requested by the engineer, POLAR prints a list of the branch currents which it can't compute.

If the equation requested by the engineer for a functional output was not supplied in EQUOUT by the engineer, POLAR prints a diagnostic.

AC	AC equivalent circuit.
COMP (201)	The number of the circuit element. (This is not used after we have finished with subroutine EXCH.)
FREQ (1, 1)	Frequency at which the circuit was solved.
ICOMP	The array of circuit element numbers in the equivalent circuit.
ICR	0 or 1. ICR = 1 if branch currents are encountered which can't be computed because the circuit element has been dropped on conversion to the equivalent circuit.
INODE (400)	The node of interest in the equivalent circuit. See appendix on Network Storage.
IPR	0 or 1. 1 if we encountered node voltages which we couldn't compute because they had been dropped on conversion to the equivalent circuit; 0 otherwise.
JCOMP	Total number of circuit elements supplied by the engineer.
JS	Number of functional outputs requested.
K	0 or 1 depending on whether we have any branch current requests (K = 1) or not (K = 0), and whether or not we have already printed our heading.
LEQU (3, 1)	The functional output equation number (in LEQU (1, I) and the title (in LEQU (2,1) and LEQU (3,1)).
LF	The index of FREQ.
NAME (201)	Up to 4 alphanumeric characters to title a circuit element.
NC	Either DC or AC depending on the type of circuit.

POLAR (Continued)

NCUR (1)	An array of 0 or 1 to indicate the circuit element number through which we want the branch current. If NCUR (5) = 1, then we want the branch current through circuit element 5.
NEQU (1)	An array of node equivalences. If node numbers were changed on conversion to the equivalent circuit, say node 7 was changed to node 4, then NEQU (8) = 4.
NODE (2, 1)	The primary (NODE (1, I)) and secondary (NODE (2, I)) nodes between which a circuit element lies in the read-in circuit.
NODMAX	The maximum number of nodes in the read-in circuit.
NOUT (1, 1)	An array of 0 or 1's. If NOUT (5) = 1, then we want the output voltage from node 5.
NTYPE (201)	The type of circuit element, either R, L, C, A, Z, E, V, D, I, F, G, B, or H.
OU (2)	Temporary storage for the real output, OU (1) and imaginary output, OU (2).
OUTPUT (2, 1)	Contain the node voltages. OUTPUT (1, 3) is the real part of node 2 output voltage. OUTPUT (2, 3) is the imaginary part of node 2 output voltage.
XMAG	Magnitude of the output.
XPHS	Phase of the output.

READFQ (NLF, FREQ, BLANK)

READFQ is called by the main program and subroutine SPECIN (when a special solution is requested) and reads in or computes the frequencies for AC nominal, special, or frequency-plot output requests. Three types of frequency constructions can be requested:

1) A list of frequencies supplied by the engineer;

2) A linear construction of frequencies over a range with equal increments;

3) A logarithmic group of frequencies constructed between a range of decades with a choice as to the number of points wanted in each decade.

A combination of types 1 and 2 or 1 and 3 can be selected, and READFQ will order the frequencies from least to greatest.

BLANK	Alphanumeric character defined as a blank in the main program.
CHECK	A read-in character that should be blank if the data cards are in the correct order.
DUMB (100)	Array used for sorting the frequencies when both read-in and constructed values are used.
FREQ (1, 1)	Array of different frequencies to be supplied to the calling program on return.
I	Index of the DUMB array when we are ordering the frequencies in a combination type of frequency.
IT 123	Read-in variable controlling whether the frequencies are to be read in (1), constructed (2), or a combination of both (3).
J	Index of the DUMB array when we are ordering the frequencies in a combination type of frequency.
K	Index of the FREQ array when we are ordering the frequencies in a combination type of frequency.
LL	0 if this is a logarithmic construction of frequencies; 1 if a linear construction.
LTYPE	0, 1, 2, or 3 and is read in when LL = 0. It controls the number of intervals minus 1 within the decades we will compute. LTYPE = 0 if LL = 1; 1 if decades only are to be used; 2 if one intermediate point at 3 is desired; and 3 if two points, at 2 and 5, are desired.
NDIF	Number of frequencies not constructed in a combination type because the final decade was reached before NLFSP frequencies were constructed.

READFQ (Continued)

NLF	Total number of frequencies to be used in solving the circuit.
NLF2	Number of frequencies to be constructed if this is a combination of the two types of frequency.
NLF3	First read-in frequency index in the DUMB array when a combination of the two types of frequencies is called for.
NLFSP	Number of frequencies to be read in if this is a combination of the two types of frequency.
NSAV	A temporary storage location for NLF so we can use the same part of the program for logarithmic and combination-type frequency construction.
START	The starting frequency value for a constructed set of frequencies.
STEP	The step size if LL = 1 (a linear construction), the final decade value if LL = 0.

READFQ calls no subroutines.

RECTAN (RV, A, B, VAL)

RECTAN is called when a Monte Carlo solution is requested and a circuit element must be varied according to a rectangular distribution. RECTAN is entered once for each such circuit element unless the element is an A or Z type, in which case RECTAN is entered twice.

A	Lower boundary of distribution.
B	Upper boundary of distribution.
RV	A uniformly distributed random variable from the main program.
VAL	The random variable obtained from subroutine RECTAN.

RECTAN calls no subroutines.

SENSPR (NODMAX, NOUT, ISN, NTYPE, NAME, NV, SV, IWHICH, JS1, JS, LEQU, OUTPUT, NODE, NOD, NEQU, XLOW, XHIGH, NUM, FREQ, NCUR, JCOMP, INODE, XNOM)

SENSPR is called by the main program when a Type Card has specified a mode 1 sensitivity output. It is called once for each circuit element that is varied in a sensitivity solution for each frequency. The main program obtains, with nominal values, a low value, and a high value node voltages before calling SENSPR. Thus any entry to SENSPR will be with three sets of solutions for the variation of one circuit element in the equivalent circuit. SENSPR tests to see what type of output is required and stores the computed output in a printing array as it is computed. When all the outputs for this circuit element, or when eight outputs (to make up a line of printing) have been calculated, SENSPR prints a line and, if the outputs have not all been printed, continues computing, storing, and printing. No attempt is made to sort or rearrange the outputs. They are printed in the order of their selection on the data cards by the engineer, with node voltages, then branch currents, then special functions.

The circuit elements are varied in the order of their appearance in the data deck with elements whose numbers (col. 1-3) were assigned by engineers being put in that number's position in the array.

SENSPR returns to the main program so that the next circuit element in the equivalent circuit can be varied.

FREQ	Frequency at which the circuit was solved.
I1	I + 1, used in NEQU because zero indices are not allowed.
IPR	First index of the SV array; when IPR = 8, we are ready to print since we can only get 8 outputs across a line of 130 characters.
IPR1	Index which addresses the first part of the labeling array NV.
IPR2	Index which addresses the second part of the labeling array NV. (NV is single dimensioned, so it has 2*IPR entries.)
ISN	Circuit element number which we have varied to obtain these solutions to the circuit.
IWHICH	1, 2, or 3 depending on whether we have a DC circuit (IWHICH = 1); an AC circuit where we want only the magnitude (IWHICH = 1); the phase (IWHICH = 2); or both magnitude and phase (IWHICH = 3).
JS	Number of special output functions requested by the engineer (JS = JS1).
JS1	Number of special output functions requested by the engineer.
K	Index used to get the nominal and low values from the COMP array.

SENSPR (Continued)

LEQU (3, 1)	The array of equation numbers (LEQU (1, I)) and labels for the special output functions resulting from solving these equations (LEQU (2, I)) and (LEQU (3, I)).
NAME (1)	The four alphanumeric characters used for labeling a component.
NEQU (1)	An array of equivalent node names. The index minus 1 of NEQU (I) is the read in node number while the value of NEQU (I) is the equivalent circuit's equivalent node number.
NOD	Alphanumeric word "NODE" used to label node voltage output.
NODE (2, 1)	The pair of nodes between which a circuit element lies between. (Needed for branch current computation.)
NODMAX	Maximum number of nodes in the read-in circuit.
NOUT (1, 1)	An array of 0 and 1 for node voltage selection. If NOUT (5, 1) = 1 then the output voltage off of node 5 is desired.
NTYPE (1)	Letter designating the type of circuit element. Used for labeling in SENSPR.
NUM (1)	Array of numbers from 1 to 50 used to label node voltage.
NV (2)	Contains a label for the outputs. For node voltage it contains "NODE" in NV (1) and the number of the node in NV (2). For branch current it contains NTYPE (ISN) and NAME (ISN), respectively; and for special functions it contains LEQU (2, I) and LEQU (3, I), respectively.
OU (2)	Temporary storage of the real (OU (1)) and imaginary (OU (2)) parts of an output.
OUTPUT (2, 1)	High values of the node voltages. Output (1, I) = the real part of the voltage of node I-1.
SV (8, 3)	Used to save three lines of output printing. The line of low outputs (outputs resulting from putting the low value of the circuit element in the circuit), the line of high outputs, and a bottom line of the low minus the high divided by the nominal output make up these three lines. Up to eight outputs per line can be printed.
XHIGH (1)	The high value of the circuit element used to obtain the node voltage in OUTPUT (I).
XLOW (1)	The low value of the circuit element, used to obtain the node voltages in COMP (101 + I).

SENSPR calls subroutines OUT, EQUOUT, BRANCH.

SENSP1 (ISN, JCOMP, NTYPE, NAME, NOD, NUM, NC, DC, LF, NODMAX, NOUT, NEQU, LEQU, IWHICH, COMP, OUTPUT, JS, ICOMP, IVALUE, XNOM, XHIGH, XLOW, NEXIT, FREQ, TITLE, TYPE, SUBTLE, NCUR, NODE, INODE, STORE)

SENSP1 is called by the main program when a node 2 sensitivity output is requested, and after the main program has completed a nominal, low, and high value solution for a particular circuit element. SENSP1 stores the circuit element's identification as well as computing up to three outputs requested by the engineer; computes the quotient of the high value minus the low value divided by the nominal value when all the circuit elements in the equivalent circuit which should have been varied have been varied; and prints a heading describing the analysis it has done and orders the quotients in STORE for one output. These are printed out and high and low worst cases are prepared by supplying the low or high value of a particular circuit element's range, depending on whether the quotient was positive or negative. After computing the two worst cases (by returning to the main program) and printing them, the next output requested and stored is ordered and printed. When all outputs and their associated worst cases have been computed and printed, a variable is set to flag this completion and SENSP1 returns to the main program.

COMP	Circuit element's number in the read-in circuit, but now used to store output voltages. COMP (I), I = 1,100 contains nominal values. COMP (I), I = 101, 200 contains low value solutions.
CUR (2)	Branch current output.
DC	DC equivalent circuit.
FREQ (1)	Frequency at which the equivalent circuit is being solved.
ICOM (200)	Circuit element's number or index in the read-in circuit. Since not all circuit elements are varied in a sensitivity analysis, and since their order is disarranged for each output, computer time is considerably lessened by saving circuit element indices in ICOM arranged in the order of the NWORST array, so that when the circuit is being prepared for a worst case analysis, it can be done so with a minimum of programming effort.
ICOMP (1)	Circuit element numbers in the equivalent circuit.
INODE (1)	One of the nodes in the equivalent circuit. See appendix on Network Storage.
IS	Index of the circuit element variation. While ISN steps from 1 to JCOMP, some circuit elements will not be varied in a sensitivity analysis and so ISN will skip values. IS does not, and serves as the middle index of the STORE array as well as an index in the ICOM, NAMECP, and NWORST arrays.
ISN	Circuit element number (of the read-in circuit) of the element currently being varied.

SENSP1 (Continued)

IVALUE (2, 1)	Present value of the circuit element in the equivalent circuit. IVALUE (1, I) contains the real part, IVALUE (2, I) the imaginary part.
IWHICH	1, 2 or 3 depending on whether we want magnitude, phase, or both, respectively, when computing and printing the outputs.
J	An index used to compute the index in the COMP array from I to L. J is the middle index when sorting and printing STORE array. It is also the index which controls the circuit element.
JC	Two times the number of circuit elements in the equivalent circuit.
JCOMP	Total number of circuit elements in the engineer's read-in circuit.
JS	Total functional outputs requested for this solution request.
K	A number from 1 to 6, used to store nominal (K = 1 or 4), low (K = 2 or 5), and high (K = 3 or 6) output solutions in STORE. K = 4, 5 and 6 is only used for phase storage when IWHICH = 3. K is also used as the circuit element's number when setting up the worst case. K is also used to obtain output in worst case analysis.
L	Used to obtain nominal and low output values from the COMP array. When L = 0 we are obtaining nominal outputs; when L = 100 we are getting low value outputs. L is also used as the IVALUE index in setting up worst cases.
LEQ	Temporary storage for LEQU (1, K), the equation number of a functional output request.
LEQU (3, 1)	The functional output equation number (LEQU (1, I)) and titles for labeling purposes (LEQU (2, I) and LEQU (3, I)).
LF	Index of the frequency array.
M	1 through 6, used for the first index in printing the STORE array.
N	The output index while printing.
NAME (1)	The circuit element's title used with NTYPE for labeling.
NAMECP (2, 200)	The labels titling the name of the circuit elements which are varied. NAMECP (1, I) = NTYPE (ISN) and NAMECP (2, I) = NAME (ISN).

SENSP1 (Continued)

NAMO (2, 5)	The labels used for titling the outputs requested. If the output is node voltage, NAMO (1, NO) = the word "NODE", and NAMO (2, NO) = a number from 1 to 50. If branch current is the output, the word "CURRENT" is contained in the two positions of NAMO. If functional output is requested, NAMO (1, NO) = LEQU (2, I) and NAMO (1, NO) = LEQU (3, I).
NC	Either AC or DC, depending on the type of equivalent circuit.
NCUR (1)	An array of 0's and 1's to indicate if a particular circuit element is to have the branch current going through it computed as an output. If NCUR (4) = 1, then the current through circuit element 4 is requested as output.
NEQU (1)	An array of equivalent node names. When we convert from the read-in circuit to an equivalent circuit, we renumber the nodes. Thus if NEQU (6) = 2, node 2 is now in the equivalent circuit.
NEXIT	-1 if this is the first time we have entered SENSP1 for this solution request card. We then initialize the output labels and set NEXIT = 0. -0 when we are storing the outputs for the varying components, but are not yet ready to print the ordered outputs. 1 when we have filled the IVALUE array with component values set to produce a worst case low output. 2 when we have the circuit element's IVALUE's filled with worst case high values. We now print both high and low worst case. 5 when the sensitivity analysis is completed.
NO	Index of the output we are on, used as the last index in the STORE array and as an index in the NWC, NAMO, and PNOM arrays.
NOD	The letters "NODE" used for labeling node voltage request prints.
NODE (2, 1)	The primary (NODE (1, I)) and secondary (NODE (2, I)) nodes of a circuit element in the read-in circuit.
NODMAX	Maximum number of nodes read into the circuit.
NOUT (1, 1)	An array of 0's and 1's. If NOUT (4) = 1, then we want to compute the output voltage from node 4.
NTYPE (1)	Type of circuit element, either A, Z, R, C, L, V, E, D, I, B, H, F, or G. Used here for labeling.
NUM (1)	A number from 1 to 50, used for labeling node voltage request prints.

SENSP1 (Continued)

NW	Temporary storage for NWC (1, N), the array which directs us to the right part of the program for the particular type of output being processed.
NWC (2, 5)	Controls the type of output being considered. Thus NWC saves program testing by directing us to the part of the program needed to compute a particular type of output. If NWC (1, 3) = 1, then the 3rd output is a node voltage kind, and if NWC (1, 3) = 2 it is a branch current output; while if NWC (1, 3) = 3, then the third output would be a functional output. The second index is a reference to which one of that type. Thus if NWC (2, 3) = 6 for NWC (1, 3) = 1, we would be requesting node voltage off the 6th node; NWC (2, 3) = 6 for NWC (1, 3) = 2 would be requesting branch current through the sixth circuit element; and NWC (2, 3) = 6 when NWC (1, 3) = 3 would be requesting a functional output employing the sixth equation of EQUOUT.
NWORST (200)	An array of 1's and 2's set to flag if the quotient is negative or positive, respectively, for this output. NWORST is set when we are sorting the quotients before printing them. It is used to control how the circuit elements are set (whether their low or high value is to be used) when we get ready to compute worst case solutions.
OUTPUT (2, 1)	The node voltage outputs. For example, OUTPUT (1, 4) contains the real part of the voltage at node 3, while OUTPUT (2, 4) contains the imaginary part of the voltage at node 3. In SENSP1, output contains high value solutions.
PNOM (10)	Contains the nominal solutions for the outputs requested. If IWHICH = 3, the phase outputs are in PNOM (J + 5) for J = 1, JS.
STORE (6, 200, 3)	The array used to store the outputs found for up to 200 variations of the circuit elements. STORE (1, I, N) contains the nominal value of the Nth output at first; then after all the computations have been made, it contains the high-minus-low divided by nominal value for the Ith component varied. STORE (2, I, N) contains the low value solutions, and STORE (3, I, N) the high value solutions. If both magnitude and phase are requested for all the outputs (if IWHICH = 3), the magnitude solutions are stored in STORE (1, I, N), STORE (2, I, N) and STORE (3, I, N); while the phase solutions are stored in STORE (4, I, N), STORE (5, I, N), and STORE (6, I, N).
SUBTLE (1)	Subtitle on the solution request card of the main program.
TITLE (1)	Title of this circuit read into the main program.
TYPE (1)	Type of solution requested--in this case, "SENS".

SENSP1 (Continued)

WCMAXM	Magnitude of the output for a worst case high solution.
WCMAXP	Phase of the output for a worst case high solution.
WCN	Worst case low output, either phase or magnitude, whichever was requested.
WCSAVIN	Worst case low phase output, saved when IWHICH = 3 for later printout.
WCSAVX	Worst case high phase output, saved when IWHICH = 3 for later printout.
WCX	Worst case high output, either phase or magnitude, whichever was requested.
XHIGH (1)	High value of the circuit element in the read-in circuit.
XLOW (1)	Low value of the circuit element in the read-in circuit.
XMAG	Magnitude of the output.
XNOM (1)	Nominal value of the circuit element in the read-in circuit.
XPHS	Phase of the output.

SENSP1 calls subroutines OUT, BRANCH, EQUOUT.

SOLUT (LF, NDEBUG, NC, TYPE, SUBTLE, ND, JNODE, MS, ITYPE, INODE, NODMAX, NERROR, V, B, E, NEQU, R, KS, NTYPE, NAME, NODE, IP, NPOW, XNOM, JCOMP, H)

SOLUT is called by the main program, once for each solution SNAP II requires, after subroutine ASSIGN has been called. SOLUT tests that the value computed in ASSIGN for a resistor is real. If a resistor was assigned an imaginary part through an equation in EQUIN, NERROR is set to 1 in ASSIGN and on testing in SOLUT, a diagnostic is printed, the circuit is printed, and the run stops.

If NERROR = 0, and the debug print option is 1, the equivalent circuit is printed with the values for the circuit elements assigned by ASSIGN.

Next, SOLUT tests for voltage sources. Any voltage source that has not already been converted to current (in which case NSCR (I) = -20) must be converted to current. To do this, SOLUT calls subroutine CURENT once for each voltage source. CURENT must be called even if this is an equivalent circuit used many times before. Since the values assigned by ASSIGN are different for each solution, the calculation of current will be different, even if the topology of the circuit changes the first time through CURENT.

Finally, the equivalent circuit is again printed out if NDEBUG = 1 and if this is a DC circuit or the first frequency of an AC circuit.

B	Voltage-dependent current source.
DUM (201)	A place holder for common storage.
E	AC voltage circuit element.
H	Voltage-dependent voltage source.
I1	A node index minus 1 used to make the resulting number equal to the correct node name, since we use zero nodes, but cannot use zero as an index.
ICOMP (400)	Circuit element numbers in the equivalent circuit.
INODE (1)	One of the nodes between which a circuit element in the equivalent circuit lies.
IP	Index used by the NPOW array.
ITYPE (1)	Circuit element type in the equivalent circuit, either R, L. C, V, E, D, I, A, Z, B, H, F, or G.
IVALUE (2, 1)	Value of the circuit element in the equivalent circuit. This is changed by CURENT when the circuit element is a voltage source.
JCOMP	Number of circuit elements in the read-in circuit.
JNODE (2, 1)	Array which serves as a table of contents for the arrays describing the equivalent circuit. See appendix on Network Storage.

<u>SOLUT</u> (Continued)

KS	Index of an array in CURENT used to store the resistor values needed to convert voltage to current.
LF	Index of the frequency we are using for this solution.
MS	0 or 1 depending on whether or not, respectively, we have a new equivalent circuit. MS is set to 1 in main after returning from subroutine SOLUT.
NAME (1)	Up to 4 alphanumeric characters used to describe the circuit element besides its type.
NB	The beginning range of a DO loop which searches through one node's entries in the equivalent circuit's arrays.
NC	Type of equivalent circuit we are solving. NC = AC or DC.
ND	Number of nodes in the equivalent circuit.
ND1	Number of nodes in the equivalent circuit plus one.
NDEBUG	1 or 0 depending, respectively, on whether or not we want intermediate debugging prints.
NEQU (1)	An array of node name equivalences between the read-in circuit (contained in the index + 1 of NEQU) and the node number of the equivalent circuit. Thus NEQU (5) = 3 means that node 4 of the read-in circuit is numbered node 3 of the equivalent circuit.
NERROR	0 or 1 depending on whether ASSIGN found that formula-computed resistor elements were real or had imaginary parts, respectively.
NF	The end of a range of a DO loop. See NB.
NODE (2, 1)	Node numbers between which this circuit element lies. NODE (1, J) is the primary node, NODE (2, J) is the secondary node.
NODMAX	Number of nodes in the read-in circuit.
NODMX1	Number of nodes in the read-in circuit + 1.
NPOW (2, 1)	The array holding resistor codes for use in converting voltage to current. NPOW (1, I) holds the circuit element number.
NSCR (402)	An array used by SOLUT and CURENT to flag those voltage entries in the equivalent circuit which have already been converted to current.
NTYPE (1)	The circuit element type, either R, C, L, A, Z, E, V, D, I, B, H, F or G.

SOLUT (Continued)

R	Resistor-type circuit element.
SUBTLE (1)	A subtitle labeling the solution request card (Type Card) and used by SOLUT to label the printout when NDEBUG = 1.
TYPE	The type of solution we are calculating. This variable is used for printing a title when NDEBUG = 1.
V	Voltage-source circuit element.
XNOM (1)	Nominal values of the circuit elements.

SOLUT calls subroutine CURENT.

SOLVE (NAD, A, NDIM, N, NER)

SOLVE is called by subroutine COMPUT once for each circuit solution required. SOLVE solves sets of simultaneous linear equations in double precision for either complex or linear systems. *

Since double precision operations on machines without double precision hardware are expensive, and since most circuit admittance matrices have many zero elements, SOLVE tests for zero elements before performing such operations. The complex arithmetic is done in the program, since FORTRAN IV compilers will not do both double precision and complex arithmetic.

To reduce round-off, each row is divided by its biggest element so that all the elements are between -1 and 1. This is a compromise since full column scaling and pivoting on the largest element would further reduce round-off, but would also increase machine time significantly.

A	Impedance matrix plus the right hand sides as an additional column.
DIV	Reciprocal of the largest element in the row. If the matrix is complex, this is the reciprocal of the magnitude, squared.
DIVI	Imaginary part of the reciprocal of a complex element a_{ii}.
DIVR	Real part of the reciprocal of a complex element a_{ii}.
II	Index used to obtain the imaginary part of the column elements.
I1	(I-1) is a DO loop which computes the column vector sum below the diagonal.
I2	Index of the column entry for the DC case. See J2.
IR	Index used to obtain the real part of the column elements.
J2	Index of the column entry for the DC case. We must skip the imaginary entries to save computer time, so J2 = 2*J-1 and will index real parts only.
JI	Index used to obtain the imaginary part of the column elements.
JR	Index used to obtain the real part of the column elements.
K	Index used to go from the second to last row up to the first row.
K1	K + 1.
K2	Index of the column entry for the DC case. See J2.

*The method used is the Crout method, described in a paper in the AIEE Transactions, 1941, Vol. 60, page 1235. Further references are Hildebrand's "Introduction to Numerical Analysis" from McGraw-Hill, 1956 page 429. The Crout method is basically the back solution of a Gauss elimination scheme.

SOLVE (Continued)

KI	Index used to obtain the imaginary part of the column elements.
KR	Index used to obtain the real part of the column elements.
N	Number of rows in matrix A.
NAD	0 if this is a real-valued matrix (i.e., if we are on a DC circuit). Otherwise, if NAD = 1, the circuit is AC and the matrix is complex.
N1	Number of columns in the augmented matrix.
N2	Twice the number of rows in A, equal to the number of real and imaginary rows.
NDIM	The first dimension of the A array, needed by the compiler and sent from the COMPUT subroutine.
NM1	The number of rows minus 1, used when we are computing from the second to last row to the first row to obtain a solution.
PI	Imaginary part of the product of a_{ik} a_{kj}.
PI1	Imaginary part of a sum used to perform complex arithmetic. We are computing $(x + iy)(u + iv)$ where $(x + iy) = a_{ik}$ and $(u + iv) = a_{kj}$.
PI2	Imaginary part of a sum used to perform complex arithmetic. We are computing $(x + iy)(u + iv)$ where $(x + iy) = a_{ik}$ and $(u + iv) = a_{kj}$
PR	Intermediate sum used in computing the new elements of a row. In the complex case, it is the real part.
PR1	Real part of a sum used to perform complex arithmetic. We are computing $(x + iy)(u + iv)$ where $(x + iy) = a_{ik}$ and $(u + iv) = a_{kj}$.
PR2	Real part of a sum used to perform complex arithmetic. We are computing $(x + iy)(u + iv)$ where $(x + iy) = a_{ik}$ and $(u + iv) = a_{kj}$.
SUM	Storage variable used to hold the sum over K of a_{ik} a_{kj}.
SUMI	Sum of the imaginary parts of the sum over k of a_{ik} a_{kj}. See SUMR.
SUMR	Sum of the real parts of the sum over k of a_{ik} a_{kj}, used in the case of an AC circuit when we are accumulating lower half column sums.
XMAX	Maximum element for each row used to scale all the elements of that row.
XMIN	Minimum element for each row used to scale all the elements of that row.

SOLVE calls no subroutines.

SPCE (RR, RI, DIST, VR, N, VI, MD)

SPCE is called by the main program when a Monte Carlo solution is required and if at least one circuit element of the equivalent circuit must be varied according to a special distribution. SPCE is called for each such circuit element.

A	Admittance-type circuit element.
DIF	A variable internal to SPCE, used to store the difference between the distribution RI or RR is close to.
DIST (2, 1)	Cumulative distribution provided by main. The abscissa is in NDIST (1, I), the ordinate in NDIST (2, I).
IOVER	Variable used only by SPCE. It is 0 when computing the real-part specially distributed random variable, and is 1 when computing the imaginary part.
MD	Number of pairs of points in DIST (supplied by the main program).
N	Circuit element type from the main program.
RI	Uniformly distributed random variable supplied by main for imaginary part of circuit element if the element is A or Z.
RR	Uniformly distributed random variable supplied by main for real part of circuit element.
RV	Uniformly distributed random variable equal to RR or RI. It is used internally in SPCE.
VAL	The specially distributed random variable which will equal either VR or VI. It is used internally in SPCE.
VI	The special distributed random variable to be used by the main program for the imaginary part of the circuit element if the element is an A or Z type.
VR	Special distributed random variable to be used by the main program for the real part of the circuit element.
XM	The slope of the interpolating points. Used only internally in SPCE.
Z	Impedance-type circuit element.

SPCE calls no subroutines.

SPECIN (ISP, NHOLD, JCOMP, TITLE, NC, TYPE, SUBTLE, XNOM, XLOW, XHIGH, NCUR, NL, SPEC, NPEC, SV, KV, NV, IS, NSPEC, NAME, NTYPE, XMCHI, XMCLO, NOUT, LEQU, JS, NODMAX, LF, DC, AC, ALL)

SPECIN is called by the main program whenever we have a special solution request. If more than one set of special values is to be solved, SPECIN is called for each such set. This subroutine reads in the frequencies (if this is an AC circuit and the first time SPECIN has been called for this Type Card), reads in node voltage, branch current, and special equation output requests (if this is the first time SPECIN has been entered for this Type Card) and reads in one set of special value designator cards. SPECIN then prints the values assigned to the variables in the equivalent circuit and returns to the main program.

AC	AC circuit.
ALL	Indicates that all the node voltages or branch currents are requested.
COMP (201)	A dummy array in SPECIN.
DC	DC circuit.
ICOMP (400)	An array of all the component numbers in the equivalent circuit.
IPR	Index of SV, KV, NV arrays. When IPR = 8, a line is printed.
IS	Index used by the NPEC and SPEC array.
ISP	Index of the particular group of special values we are processing. The first time control enters SPECIN for this Type Card, ISP = 1.
IVALUE	Values for the circuit elements in the equivalent circuit.
JCOMP	The number of circuit elements in the read-in circuit.
JS	The number of entries in LEQU (i. e., the number of functional outputs designated on the Type Card).
K1	Used to find an equivalent circuit index corresponding to a particular circuit element.
K2	The other equivalent circuit index, different from K1.
KV (8, 1)	Fixed-point equivalent array of SV.
LEQU (3, 1)	An array of special outputs.
LF	Frequency index.
M	A temporary storage for the NHOLD entry.
NAME (201)	Up to 4 alphanumeric characters used to describe the circuit element besides its type.

SPECIN (Continued)

NB	A temporary storage variable for a value of NBR.
NBR (19)	A holding array for reading in individual branch current requests.
NC	AC or DC.
NCUR (1)	An array of 0 or 1 indicating branch current requests. If NCUR (6) = 1, for example, the current through circuit element 6 is desired as output.
NHOLD (1)	An array of integers from 0 to 9 which directs what value is to go in a particular component. If NHOLD (8) = 2, then circuit element 8 should take on the high value found in columns 51-60 of the Circuit Element Card. The numbers and their values are as follows: NHOLD (I) = 0, nominal value (col. 31-40). NHOLD (I) = 1, low value (col. 41-50). NHOLD (I) = 2, high value (col. 51-60). NHOLD (I) = 3, Monte Carlo low (col. 61-70). NHOLD (I) = 4, Monte Carlo high (col. 71-80). NHOLD (I) = 5, 2nd card, col. 31-40. NHOLD (I) = 6, 2nd card, col. 41-50. NHOLD (I) = 7, 2nd card, col. 51-60. NHOLD (I) = 8, 2nd card, col. 61-70. NHOLD (I) = 9, 2nd card, col. 71-80. NHOLD is read in by SPECIN on up to 3 cards (depending on the value of JCOMP). Each column of the cards is an entry in NHOLD.
NL	1 or 0 depending, respectively, on whether or not we have a nonlinear circuit.
NODMAX	Number of nodes in the read-in circuit.
NOUT (1)	An array of 0 or 1 giving the node voltage requested. If NOUT (4) = 1, then we want the voltage off of node 4.
NPEC (6, 1)	The SPEC array in fixed format, used to hold the circuit element's number in SPEC (1, I).
NSPEC (201)	An array which tells the program if this circuit element has special values in addition to those in columns 41-80 of the Circuit Element Card.
NST	A temporary storage variable for a value of NSUT.
NSUT (25)	A holding array for reading in individual node voltage requests.
NTYPE (201)	The circuit element type, either R, C, L, A, Z, E, V, D, I, B, H, F, or G.
NV (1)	The code number from the NHOLD array. This is printed out with the name and values of the circuit element.

SPECIN (Continued)

SPEC (6, 1)	The NPEC array holding special values for some of the circuit elements.
SUBTLE (1)	A title read in with the Type Card, used to title this solution.
SV (8, 1)	Used to accumulate a line of circuit element values for printing. Only 8 circuit element's values can be printed in a line. SV (I, 3) contains the numerical value, while SV (I, 1) = the type of circuit element and SV (I, 2) = its name. The arguments SV (I, 1) and SV (I, 2) are addressed as KV.
TITLE (1)	Title of the circuit, used for labeling.
TYPE	The letters "SPCL", used for labeling.
XHIGH (1)	High value of the circuit element for sensitivity analysis, or a special value.
XLOW (1)	Low value of the circuit element for sensitivity analysis, or a special value.
XMCHI (201)	Standard deviation of the Monte Carlo distribution (if a normal or lognormal distribution is required), or a special value.
XMCLO (201)	Mean of the Monte Carlo distribution (if a normal or lognormal distribution is required), or a special value.
XNOM (1)	Nominal values of the circuit elements.

SPECIN calls no subroutines.

STAT (ISTAT, JS, IW, HIHIST, TITLE, NC, TYPE, SUBTLE, LEQU, IR, NORV, NSTR, OUTPUT, XLHIST, FREQ)

STAT is called by the main program, once for each solution required, when a Monte Carlo solution is needed. Thus if NORV = 500, subroutine STAT is called 500 times.

The first time STAT is called, ISTAT is zero, which flags the program to set arrays to zero and compute the bin size. Each output can be thought of falling into a particular box, and each box size is dependent on the size of BIN. After initialization, ISTAT is set to 1 and STAT begins the part of the program common to all entries in STAT.

The output is computed for up to five outputs and the resulting number is compared to the limits read in or computed in the main program. If the output falls outside these limits, it is printed as an out of limit output and a counter is increased by one. If any output's counter exceeds 20 percent of NORV (the number of solutions for this Monte Carlo request), a diagnostic and the incomplete histogram of all the outputs are printed.

If the outputs are within the allowable range, STAT adds their values into the sums used for computing the mean and variance, and then finds what box they belong in for the histogram plot routine.

For example, if an output's value is 26.3 and the bin size (computed from the range divided by 100) is 5.1, then 26.3 would fall in the fourth box, if the boxes were though of as ranging as follows:

Box 1: 10.2 to 15.3
Box 2: 15.3 to 20.4
Box 3: 20.4 to 25.5
Box 4: 25.5 to 30.6
etc.

Thus the entry in NBOX (4, I) would be increased by one if this were the Ith output.

When IR = NORV (we have done the last solution required), then we compute the mean, variance standard deviation, and the values between which the median lies. We also rescale the bin size so it is either one-half a standard deviation (if NORV is less than 300), or one-fourth a standard deviation (if NORV is greater than 300).

Finally, the labeling used by subroutine HIST is constructed and subroutine HIST is called. On return from HIST, the mean, median, variance, and standard deviation are printed and the intervals of the 12 (if NORV is less than 300) or 24 (if NORV is greater than 300) boxes are printed. Control returns to the main program.

BEG	Temporary storage.
BIN (5)	Size of the division of the boxes for the histogram before it is scaled.
DIF	Temporary storage.

STAT (Continued)

FINISH	Incremented value used to print the intervals in JBOX after calling HIST.
FREQ	Frequency this equivalent circuit is being solved for.
FS	Temporary storage used to compute increments of standard deviation fractions.
HIHIST (5)	Upper limit of the outputs. This is read into main or computed as 20 percent of the nominal value of the output if the engineer didn't supply limits.
IK	Index of JBOX.
IM	Index of the SINGO array.
IR	Index of the solution being sought. IR = NORV when we are finished.
ISTAT	0 the first time STAT is entered for this Type Card. ISTAT = 1 thereafter. This flags the program to initialize the arrays.
IT	Used to compute entries in JBOX.
IW (5)	Array of 1 or 2's for AC outputs. If IW (I) = 1, the Ith output will be in magnitude. If IW (I) = 2 the Ith output will be in phase.
J3	Counter in printing the intervals in JBOX after calling HIST.
JBOX (50)	The NBOX array is compressed into the JBOX array for use by HIST. JBOX is divided along fractions of standard deviations.
JS	Number of outputs requested.
JT	Temporary counter.
K	Counter in printing the intervals in JBOX after calling HIST.
KB	Starting value in a DO loop to compute NBETWH.
KM	Temporary storage.
KS	Used to scale NBOX to JBOX.
LABEL (25)	Labeling array needed to label the bottom of the histogram plot.
LEQU (3, 5)	The functional output equation number (LEQU (1, I)) and titles for labeling purposes (LEQU (2, I) and LEQU (3, I)).
LIG3	2*NF used to compute JBOX entries.
M	Used to print the list of the number in each JBOX.

STAT (Continued)

M2	Temporary storage.
NB	Used to compute entries in JBOX.
NBETWL (5)	Number of outputs above the lower limit but below the minus-three sigma point.
NBETWH	Number of outputs below the upper limit but above the plus-three sigma point.
NBOX (100, 5)	An array of divisions into which we add one if the output falls into its range.
NC	Either AC or DC, depending on the type of equivalent circuit.
NF	The part of the calculation of the variable used for the upper limit of a DO loop.
NF1	The upper range of a DO loop.
NM1	Temporary storage.
NM2	Temporary storage.
NORV	The total number of solutions we will do for this Type Card.
NOUND (5)	A counter of out-of-range outputs.
NOUNHI (5)	Number of outputs above the upper limit.
NOUNLO (5)	Number of outputs below the lower limit.
NOW	Temporary counter.
NR (5)	Number of outputs within the limits used in histogram.
NRM	NR/2 used in computing the median.
NRT	Temporary storage for NR.
NSUM	Temporary storage when computing the median.
NT	Temporary counter.
NT1	2*NF used to compute JBOX entries.
OUTPUT	Node voltage output.
RANGE	Expected range of the output, computed from the limits.
RN	Floating-point value of NR.

STAT (Continued)

RN1	RN-1.
SIG	Fractional part of the standard deviation, either one-half or one-fourth of it.
SIGNO (7)	Labeling array used by HIST which contains the mean and multiples of the standard deviation.
SQUAR	Temporary storage while we are computing the variance.
STAN (5)	Standard deviation of the outputs.
STR	The starting value for the random number generator, printed in STAT.
SUBTLE (35)	Subtitle on the solution request card of the main program.
THIS	A sum when we are finding the right NBOX entry.
TITLE (40)	Title of this circuit read into the main program.
TYPE	The type of solution requested. In this case a "MONT" one.
VARI (5)	Variance of the outputs.
XINT	Sum when scaling NBOX.
XLHIST (5)	Lower bounds for the outputs. See HIHIST.
XMEAN (5)	The sum of the outputs as they are computed; used to hold the means of the outputs when IR = NORV.
XMED (2, 5)	The two box limits on the medians of the outputs.
XP (5)	The phase of the output.

STAT calls subroutines EQUOUT, OUT, HIST.

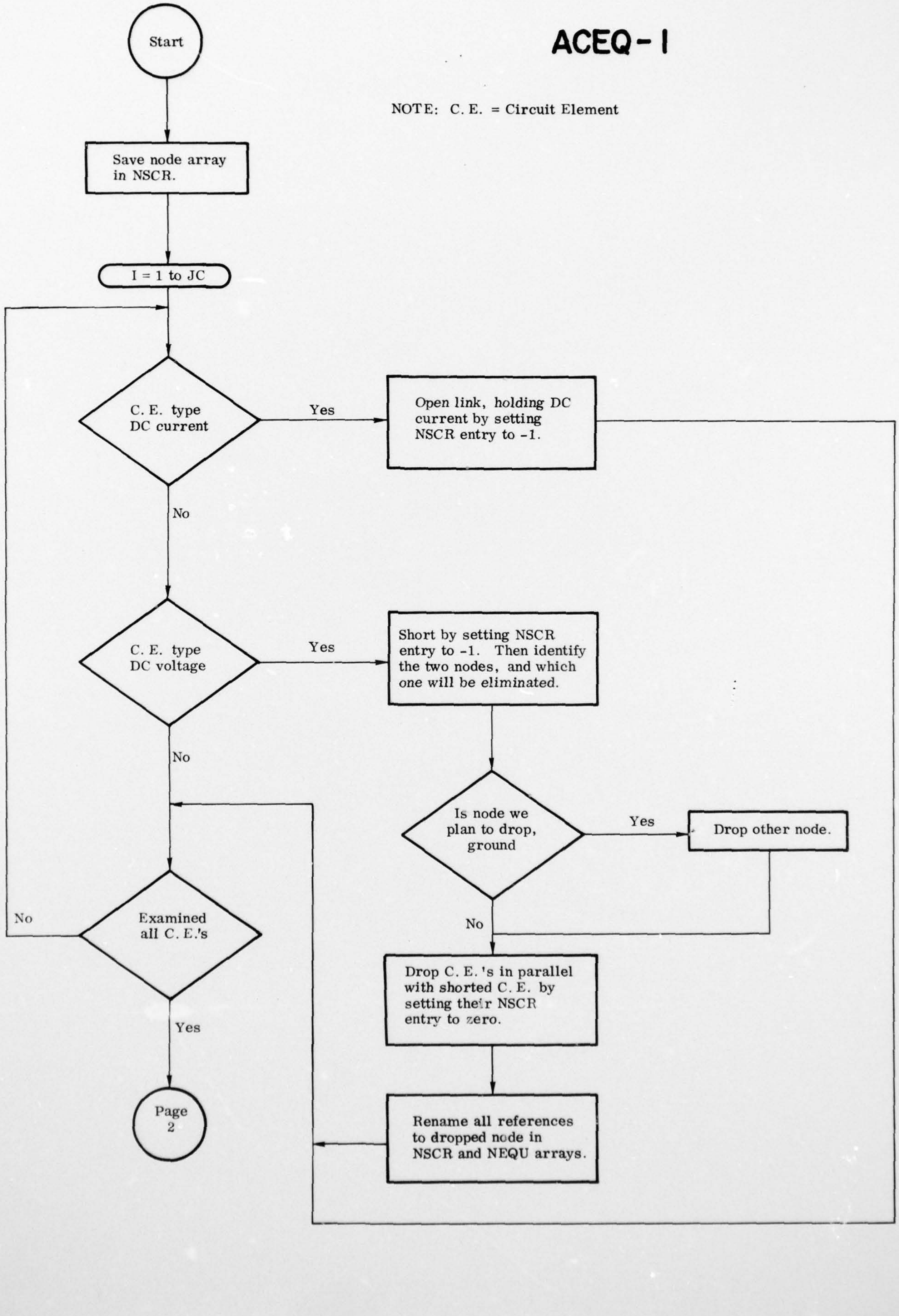
4. LOGICAL FLOW DIAGRAMS

Logical flow diagrams for the various subroutines appear in the following sequence:

ACEQ	EQUIN	READFQ
ASSIGN	EQUOUT	RECTAN
BRANCH	EXCH	SENSPR
CARD	HIST	SENSP1
CHECK	INOUT	SOLUT
COMPUT	NOLIN	SOLVE
CONECT	NORM	SPCE
CURENT	OUT	SPECIN
CURR	PLOTF	STAT
DCEQ	POLAR	

ACEQ-1

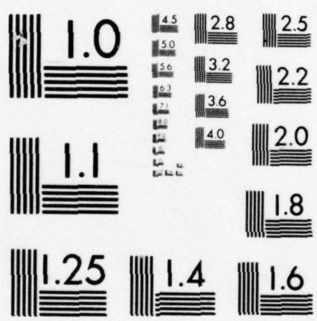
NOTE: C. E. = Circuit Element



AD-A054 694 ARINC RESEARCH CORP SANTA ANA CALIF WESTERN DIV F/G 9/5
PROGRAMMER'S MANUAL FOR SNAP II COMPUTER PROGRAM.(U)
AUG 68 T D PRICE, C KIMME
UNCLASSIFIED 474-01-2-916 NL

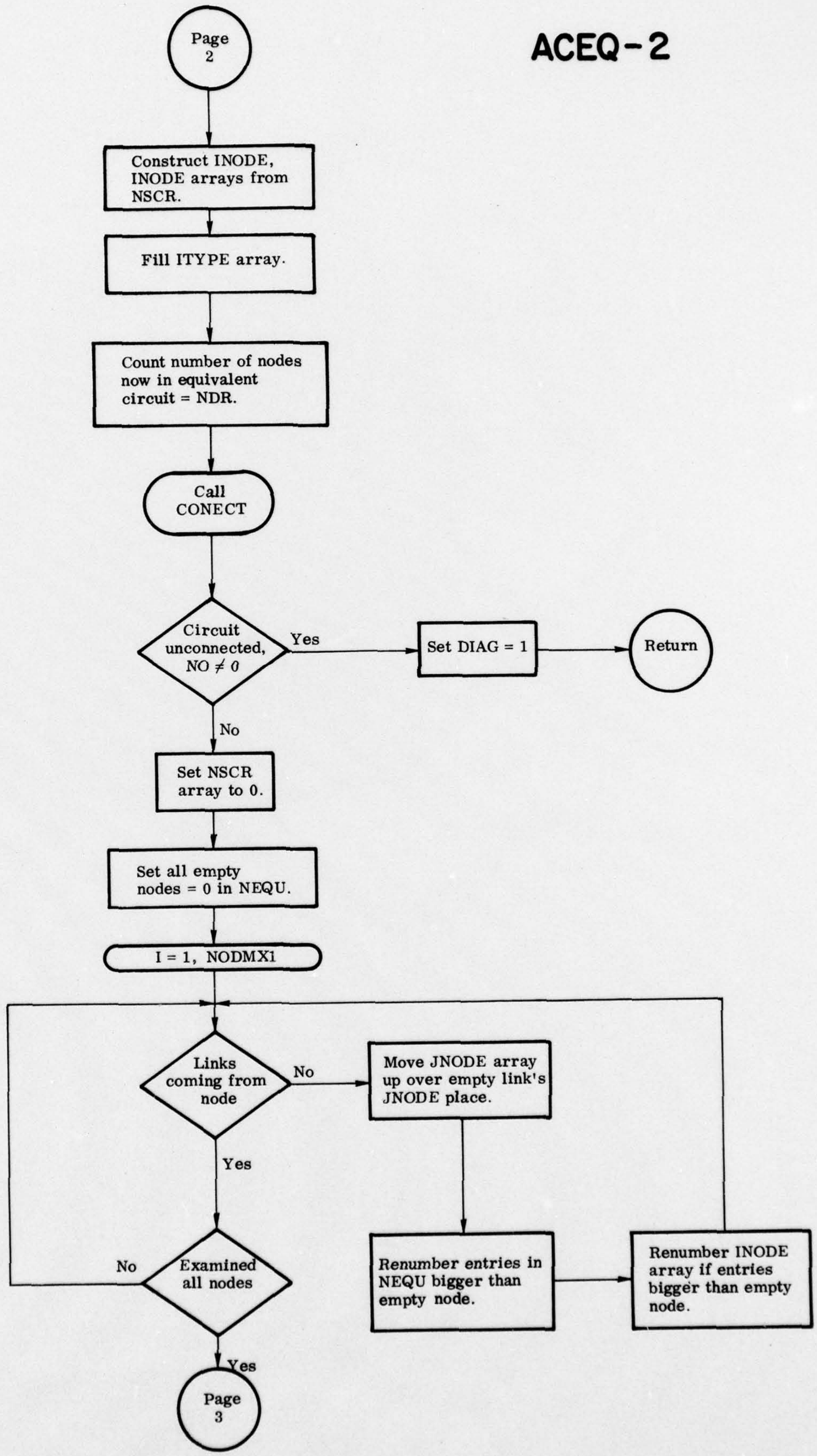
2 OF 2
AD
A054694

END
DATE
FILMED
6-78
DDC



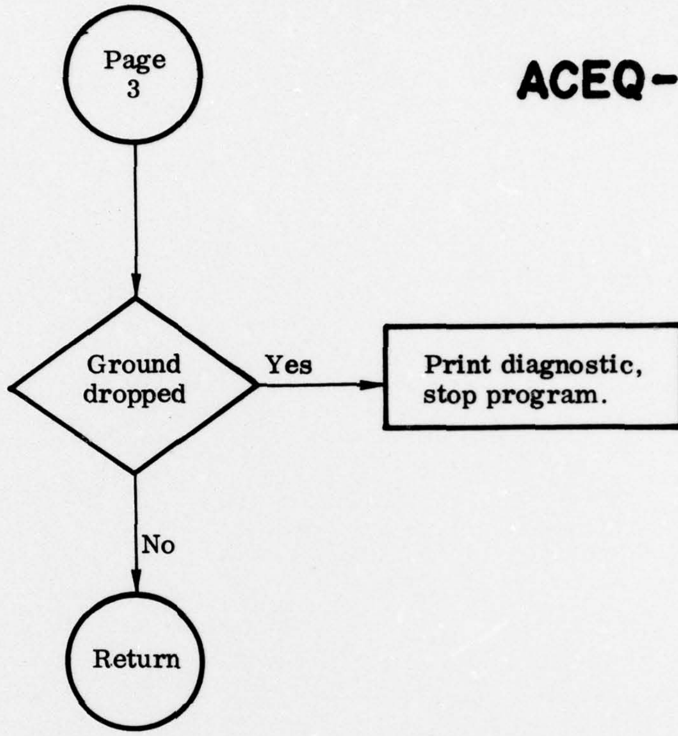
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ACEQ-2



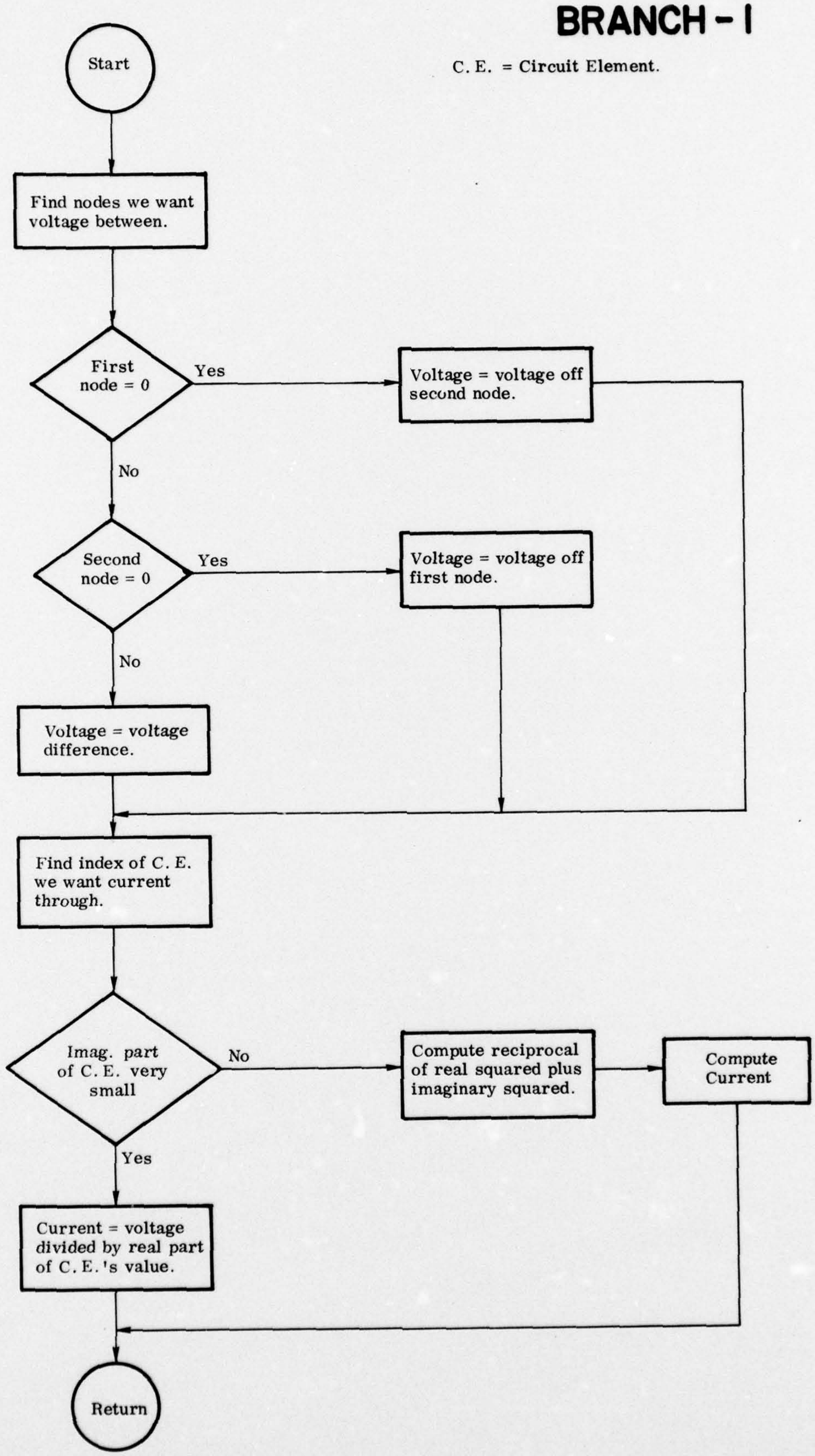
Page 2
Construct INODE, INODE arrays from NSCR.
Fill ITYPE array.
Count number of nodes now in equivalent circuit = NDR.
Call CONECT
Circuit unconnected, NO ≠ 0
Yes
Set DIAG = 1
Return
No
Set NSCR array to 0.
Set all empty nodes = 0 in NEQU.
I = 1, NODMX1
Links coming from node
No
Move JNODE array up over empty link's JNODE place.
Yes
Renumber entries in NEQU bigger than empty node.
Renumber INODE array if entries bigger than empty node.
No
Examined all nodes
Yes
Page 3

ACEQ-3

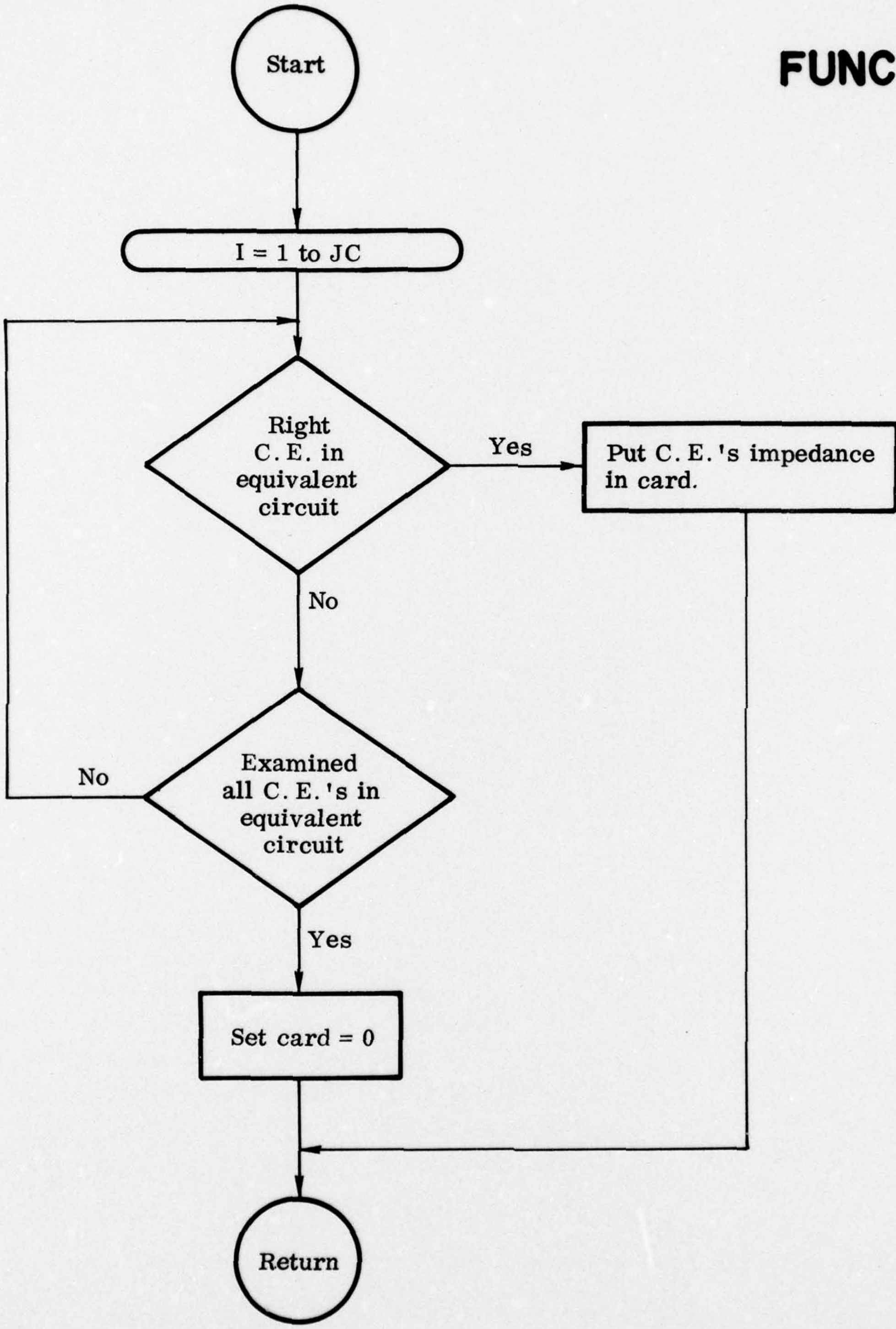

Page
3
Ground
dropped
Yes
Print diagnostic,
stop program.
No
Return

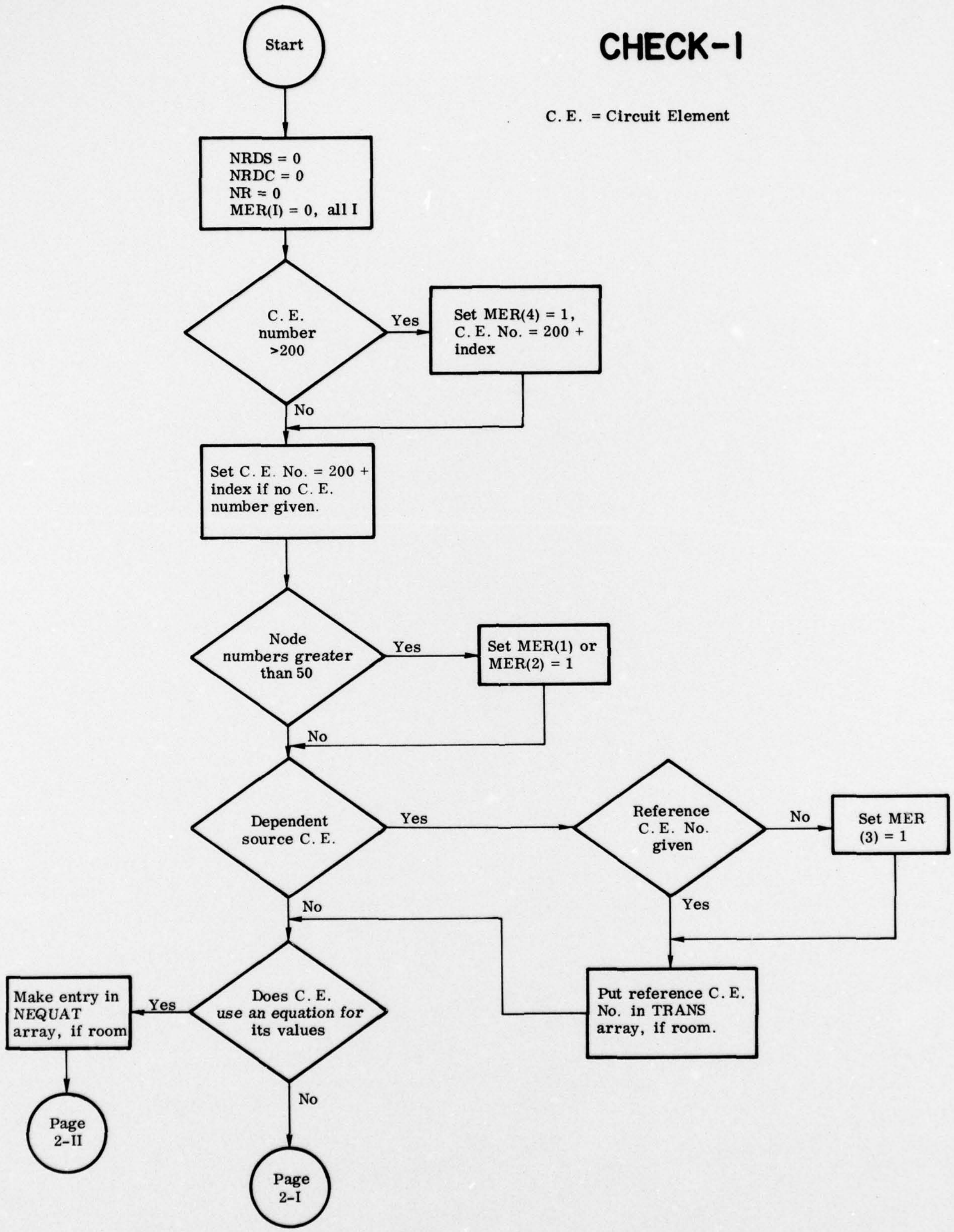
BRANCH - I

C. E. = Circuit Element.

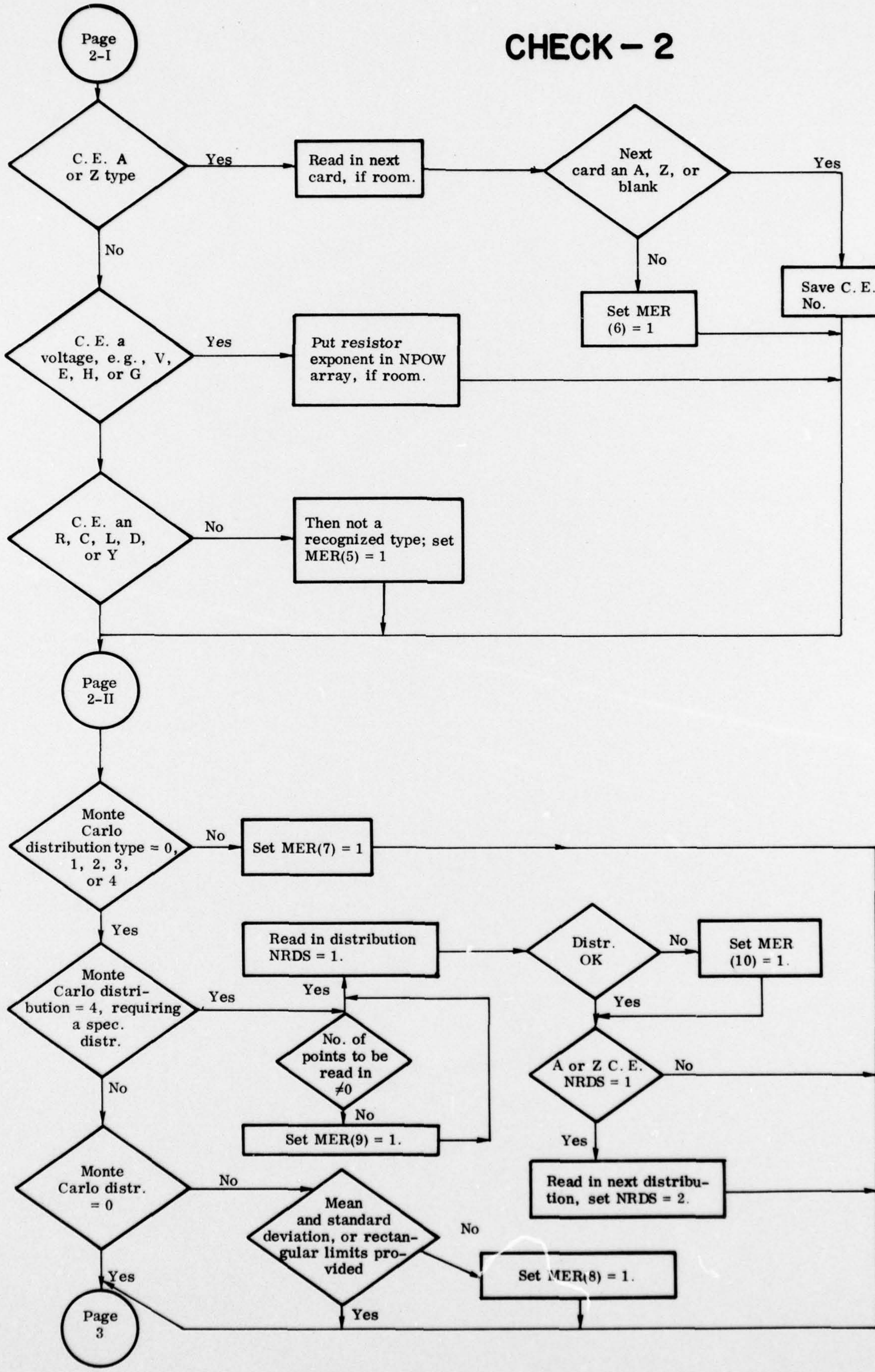


FUNCTION CARD



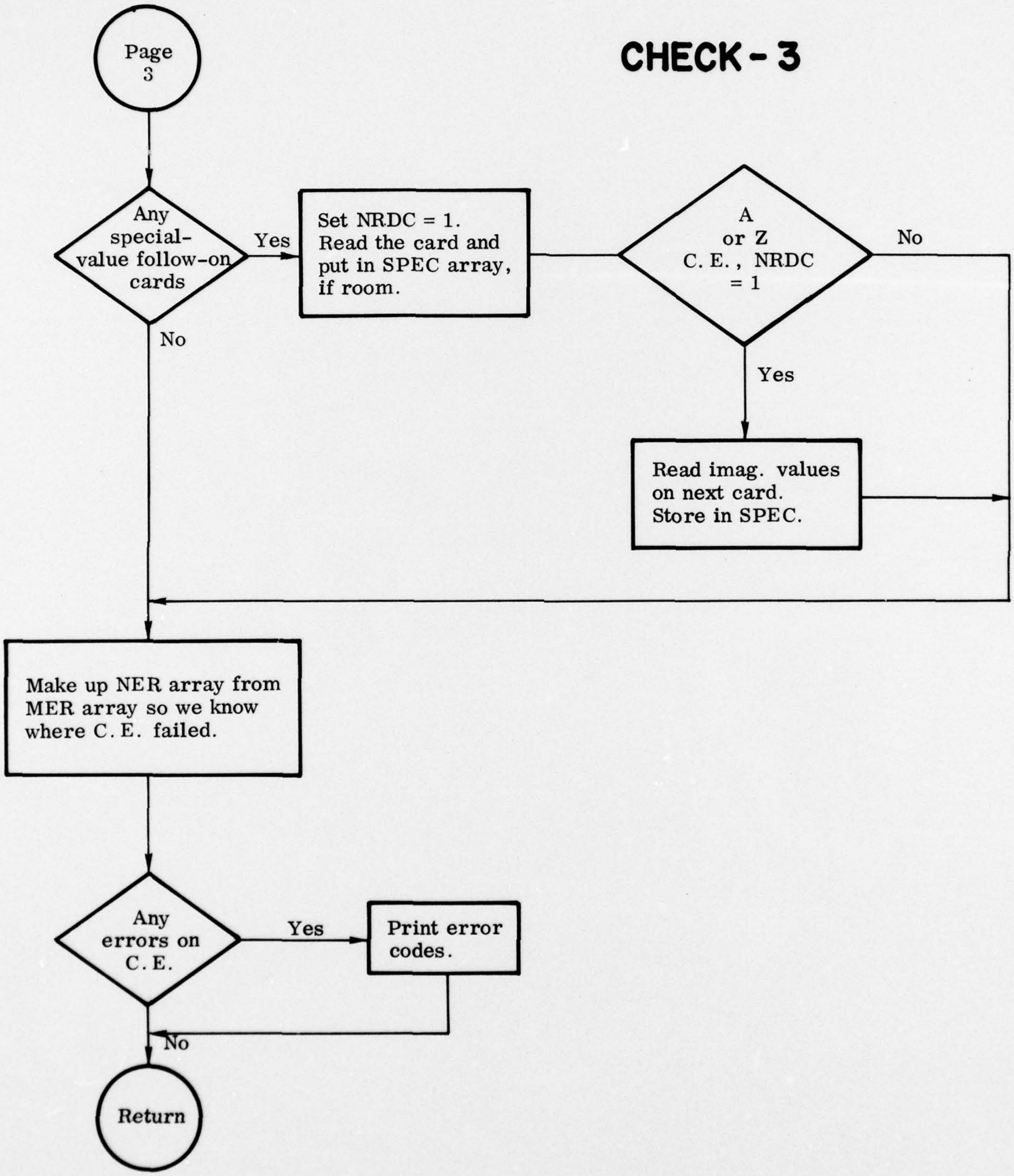

CHECK-I
C. E. = Circuit Element
Start
NRDS = 0
NRDC = 0
NR = 0
MER(I) = 0, all I
C. E. number >200
Yes
Set MER(4) = 1, C. E. No. = 200 + index
No
Set C. E. No. = 200 + index if no C. E. number given.
Node numbers greater than 50
Yes
Set MER(1) or MER(2) = 1
No
Dependent source C. E.
Yes
Reference C. E. No. given
No
Set MER (3) = 1
Yes
Put reference C. E. No. in TRANS array, if room.
No
Does C. E. use an equation for its values
Yes
Make entry in NEQUAT array, if room
Page 2-II
No
Page 2-I

CHECK - 2



Page 2-I
C.E. A or Z type
Yes
Read in next card, if room.
Next card an A, Z, or blank
Yes
No
Save C.E. No.
Set MER (6) = 1
No
C.E. a voltage, e.g., V, E, H, or G
Yes
Put resistor exponent in NPOW array, if room.
C.E. an R, C, L, D, or Y
No
Then not a recognized type; set MER(5) = 1
Page 2-II
Monte Carlo distribution type = 0, 1, 2, 3, or 4
No
Set MER(7) = 1
Yes
Read in distribution NRDS = 1.
Distr. OK
No
Set MER (10) = 1.
Yes
Monte Carlo distribution = 4, requiring a spec. distr.
Yes
Yes
No. of points to be read in ≠0
No
Set MER(9) = 1.
A or Z C.E. NRDS = 1
No
Yes
Read in next distribution, set NRDS = 2.
No
Monte Carlo distr. = 0
No
Mean and standard deviation, or rectangular limits provided
No
Set MER(8) = 1.
Yes
Yes
Page 3

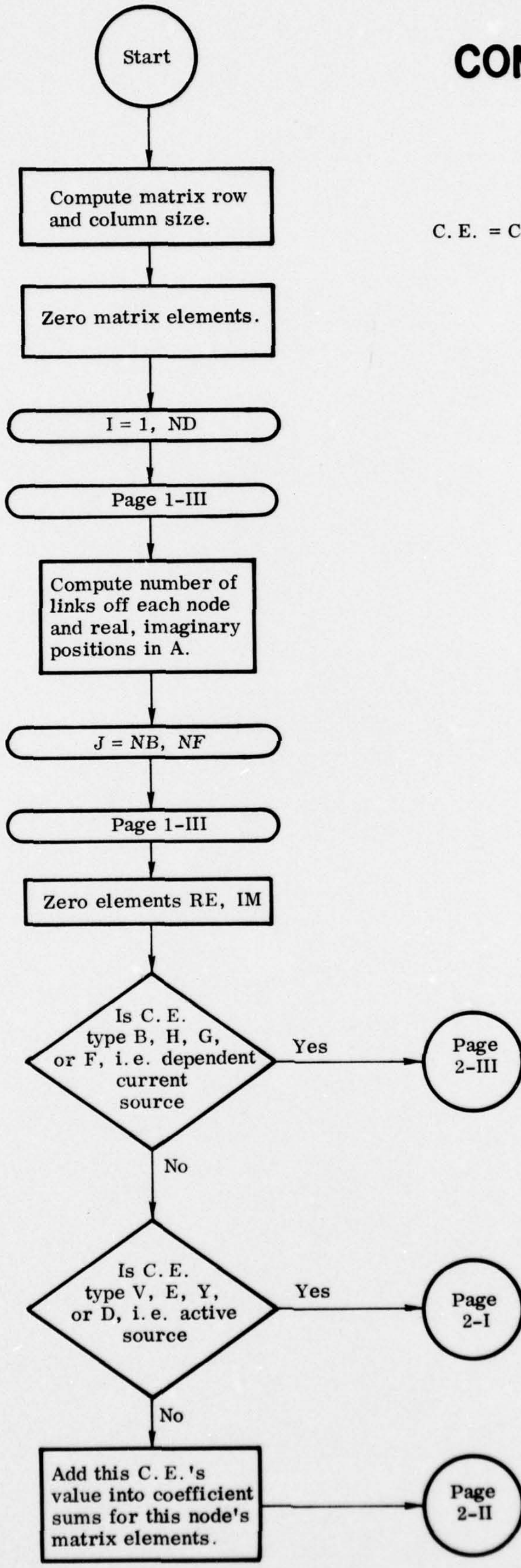
CHECK - 3



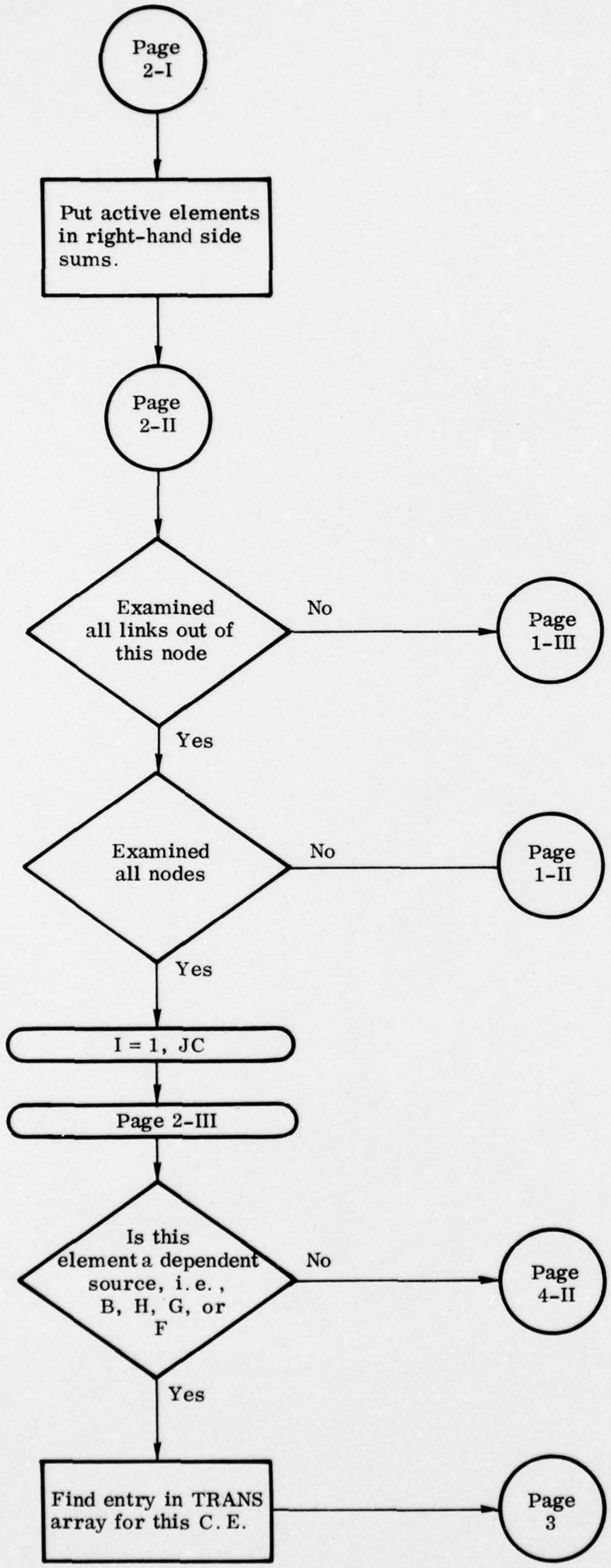
Page 3
Any special-value follow-on cards
Yes
No
Set NRDC = 1. Read the card and put in SPEC array, if room.
A or Z C. E. , NRDC = 1
No
Yes
Read imag. values on next card. Store in SPEC.
Make up NER array from MER array so we know where C. E. failed.
Any errors on C. E.
Yes
Print error codes.
No
Return

COMPUT-I

C. E. = Circuit Element.

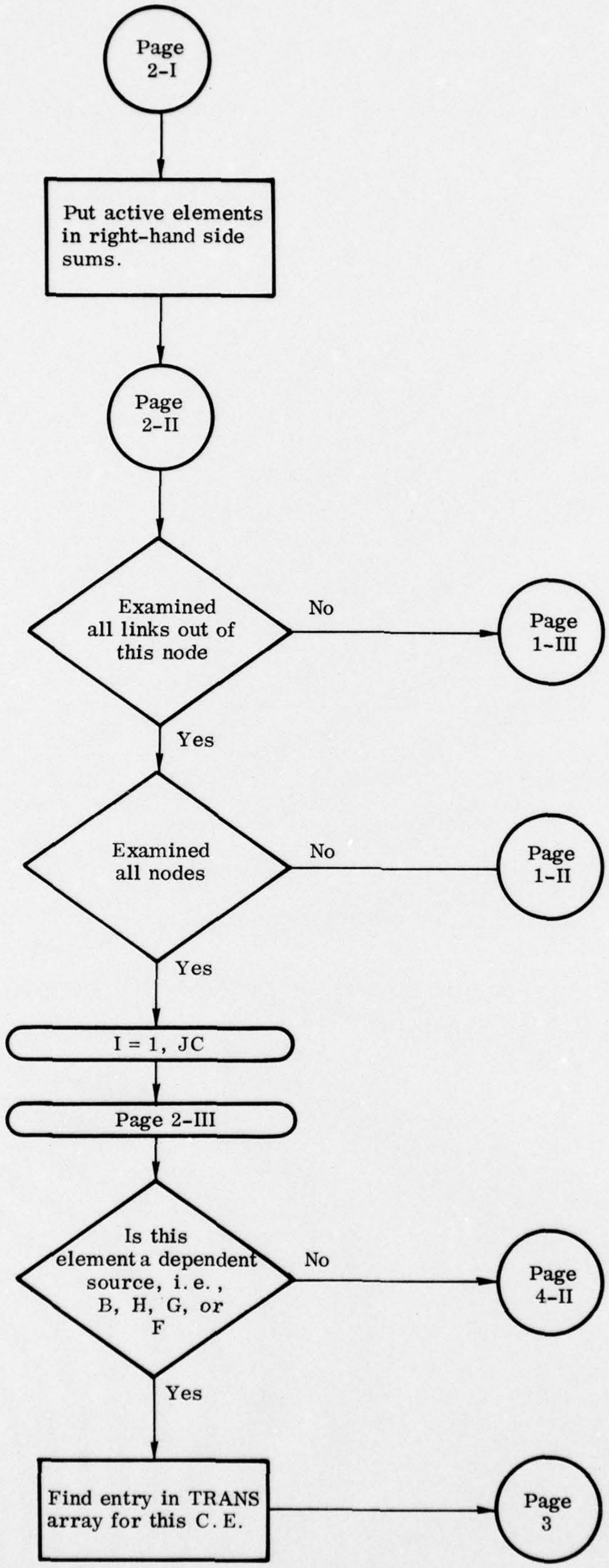


COMPUT-2

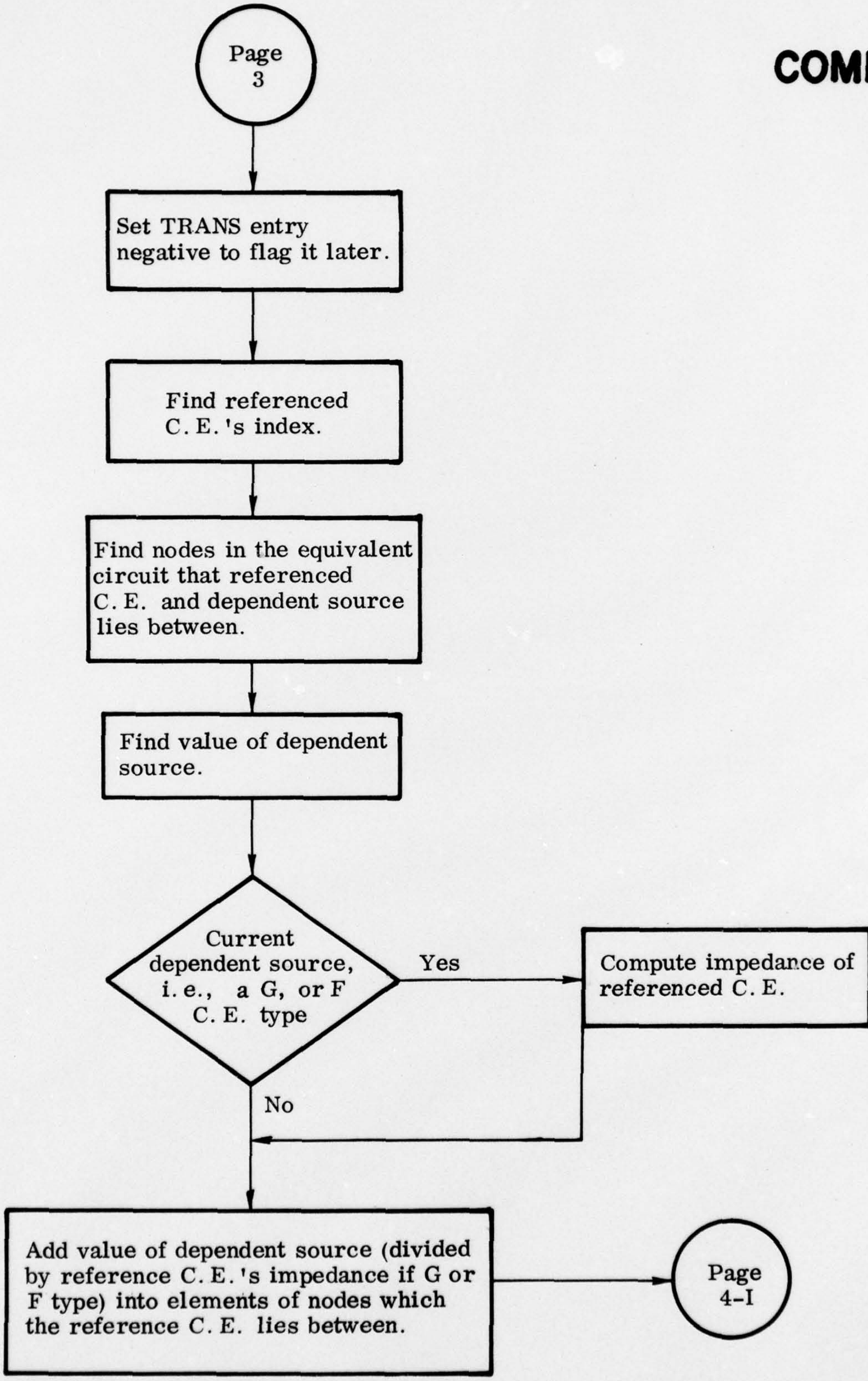


Page 2-I
Put active elements in right-hand side sums.
Page 2-II
Examined all links out of this node
No
Page 1-III
Yes
Examined all nodes
No
Page 1-II
Yes
I = 1, JC
Page 2-III
Is this element a dependent source, i.e., B, H, G, or F
No
Page 4-II
Yes
Find entry in TRANS array for this C.E.
Page 3

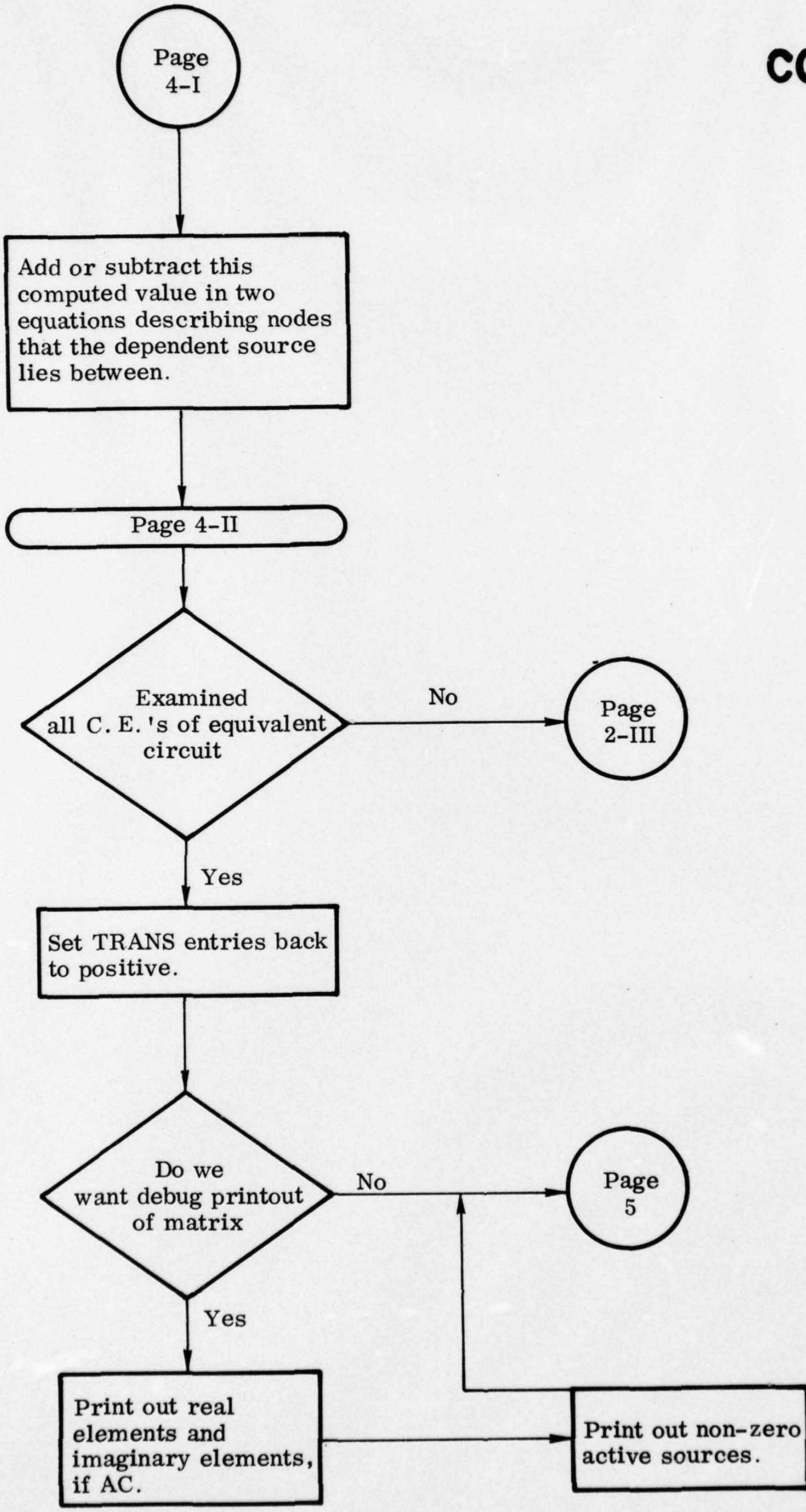
COMPUT-2


Page 2-I
Put active elements in right-hand side sums.
Page 2-II
Examined all links out of this node
No
Page 1-III
Yes
Examined all nodes
No
Page 1-II
Yes
I = 1, JC
Page 2-III
Is this element a dependent source, i.e., B, H, G, or F
No
Page 4-II
Yes
Find entry in TRANS array for this C.E.
Page 3

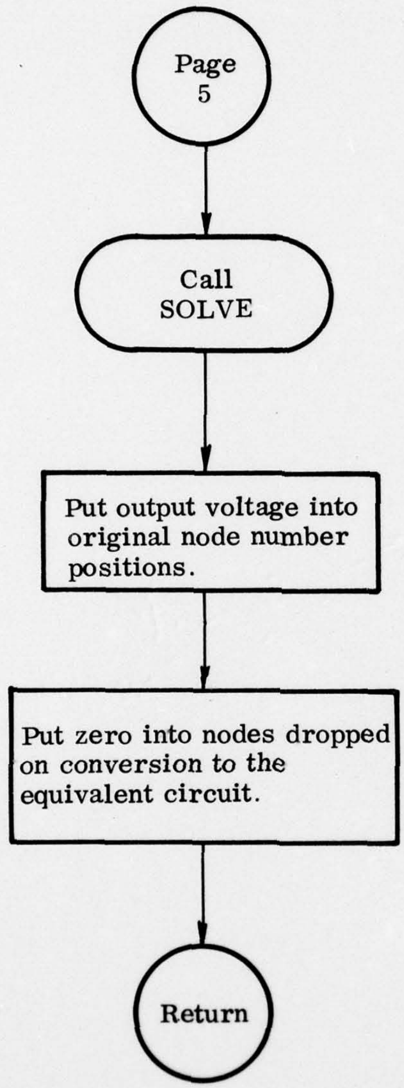
COMPUT - 3


Page
3
Set TRANS entry negative to flag it later.
Find referenced C. E.'s index.
Find nodes in the equivalent circuit that referenced C. E. and dependent source lies between.
Find value of dependent source.
Current dependent source, i.e., a G, or F C. E. type
Yes
Compute impedance of referenced C. E.
No
Add value of dependent source (divided by reference C. E.'s impedance if G or F type) into elements of nodes which the reference C. E. lies between.
Page
4-I

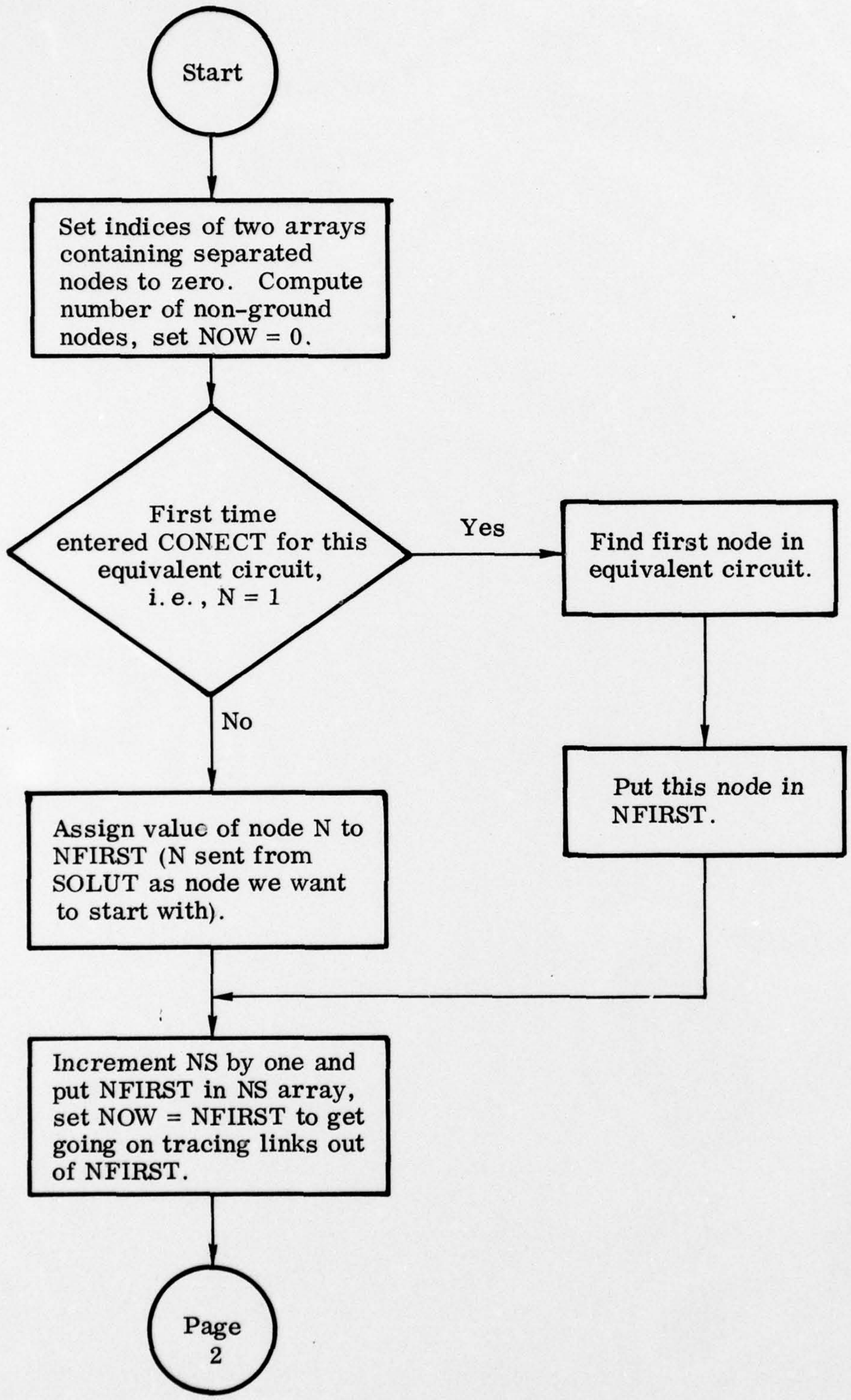
COMPUT-4


Page 4-I
Add or subtract this computed value in two equations describing nodes that the dependent source lies between.
Page 4-II
Examined all C. E.'s of equivalent circuit
No
Page 2-III
Yes
Set TRANS entries back to positive.
Do we want debug printout of matrix
No
Page 5
Yes
Print out real elements and imaginary elements, if AC.
Print out non-zero active sources.

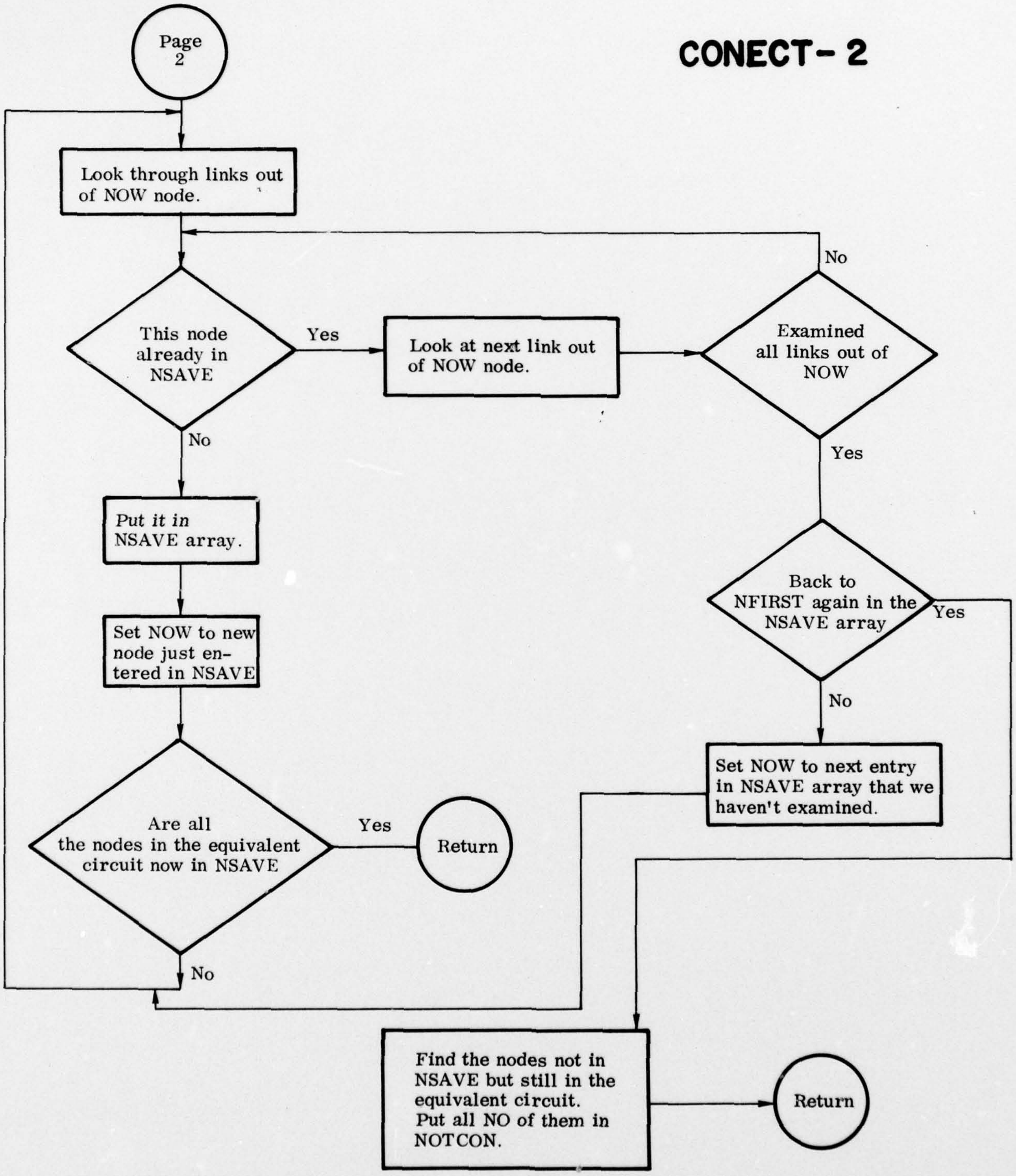
COMPUT-5


Page
5
Call
SOLVE
Put output voltage into original node number positions.
Put zero into nodes dropped on conversion to the equivalent circuit.
Return

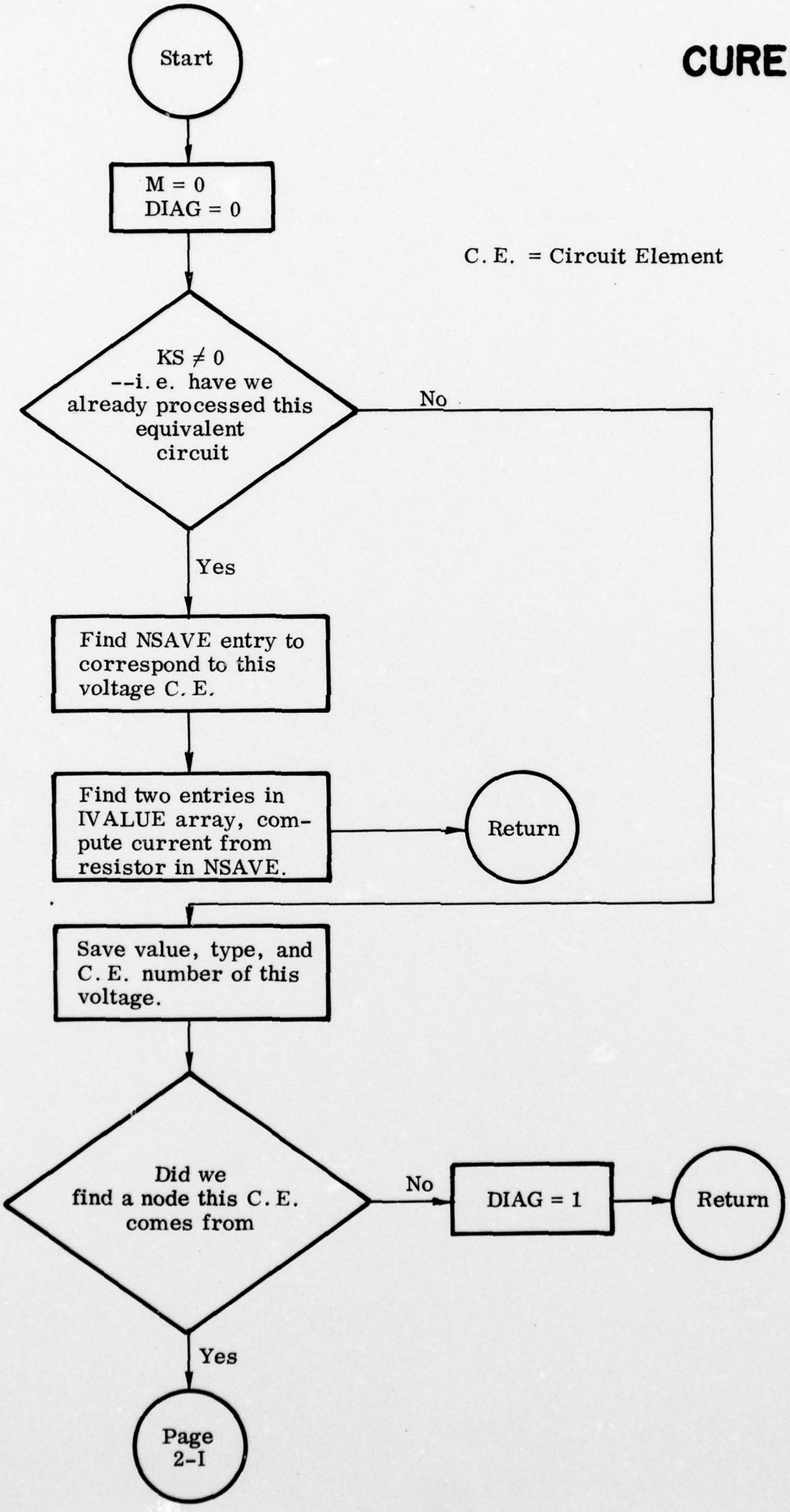
CONECT-1


Start
Set indices of two arrays containing separated nodes to zero. Compute number of non-ground nodes, set NOW = 0.
First time entered CONECT for this equivalent circuit, i.e., N = 1
Yes
Find first node in equivalent circuit.
Put this node in NFIRST.
No
Assign value of node N to NFIRST (N sent from SOLUT as node we want to start with).
Increment NS by one and put NFIRST in NS array, set NOW = NFIRST to get going on tracing links out of NFIRST.
Page 2

CONECT- 2

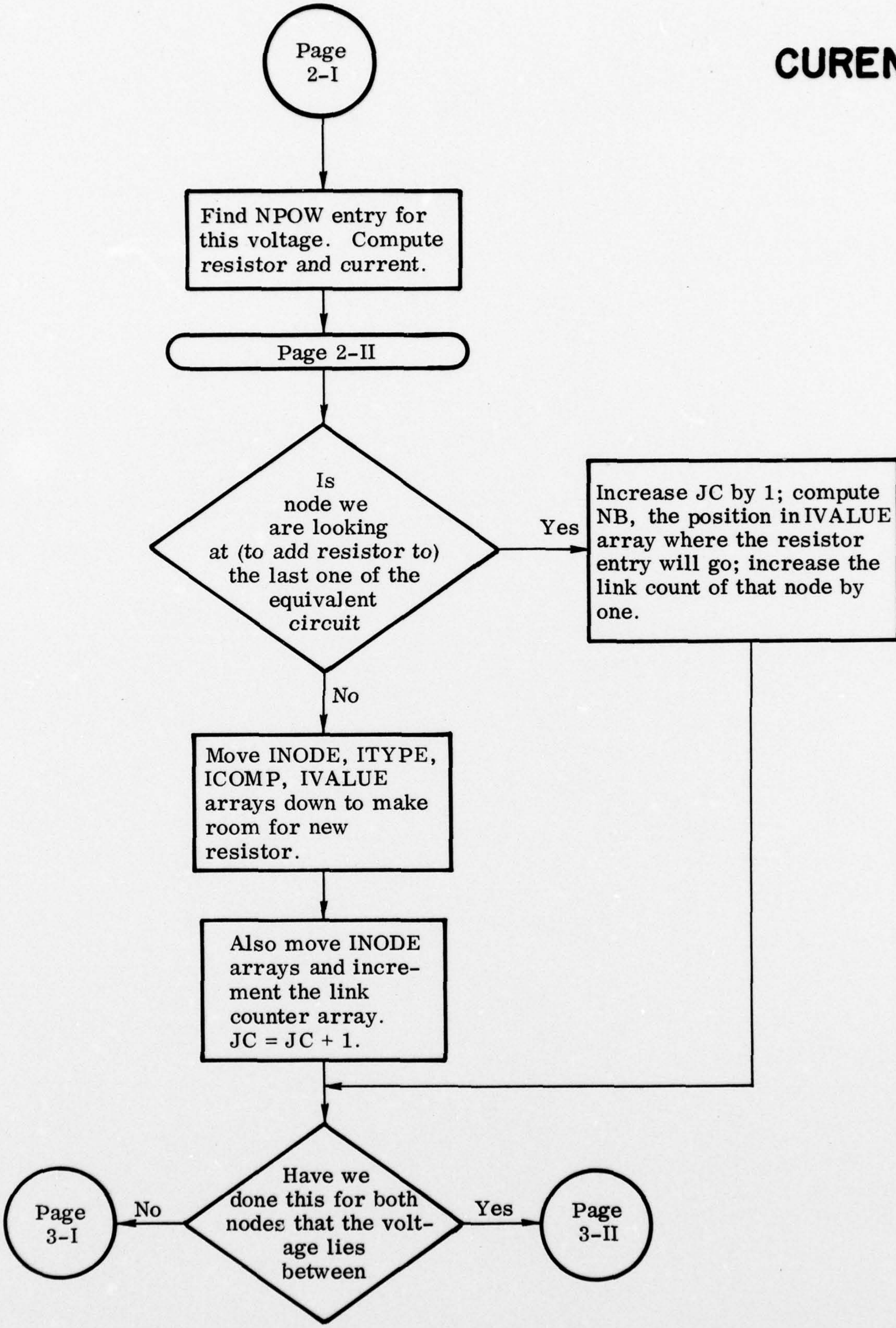

Page 2
Look through links out of NOW node.
This node already in NSAVE
Yes
Look at next link out of NOW node.
Examined all links out of NOW
No
Yes
No
Put it in NSAVE array.
Set NOW to new node just entered in NSAVE
Back to NFIRST again in the NSAVE array
Yes
No
Set NOW to next entry in NSAVE array that we haven't examined.
Are all the nodes in the equivalent circuit now in NSAVE
Yes
Return
No
Find the nodes not in NSAVE but still in the equivalent circuit. Put all NO of them in NOTCON.
Return

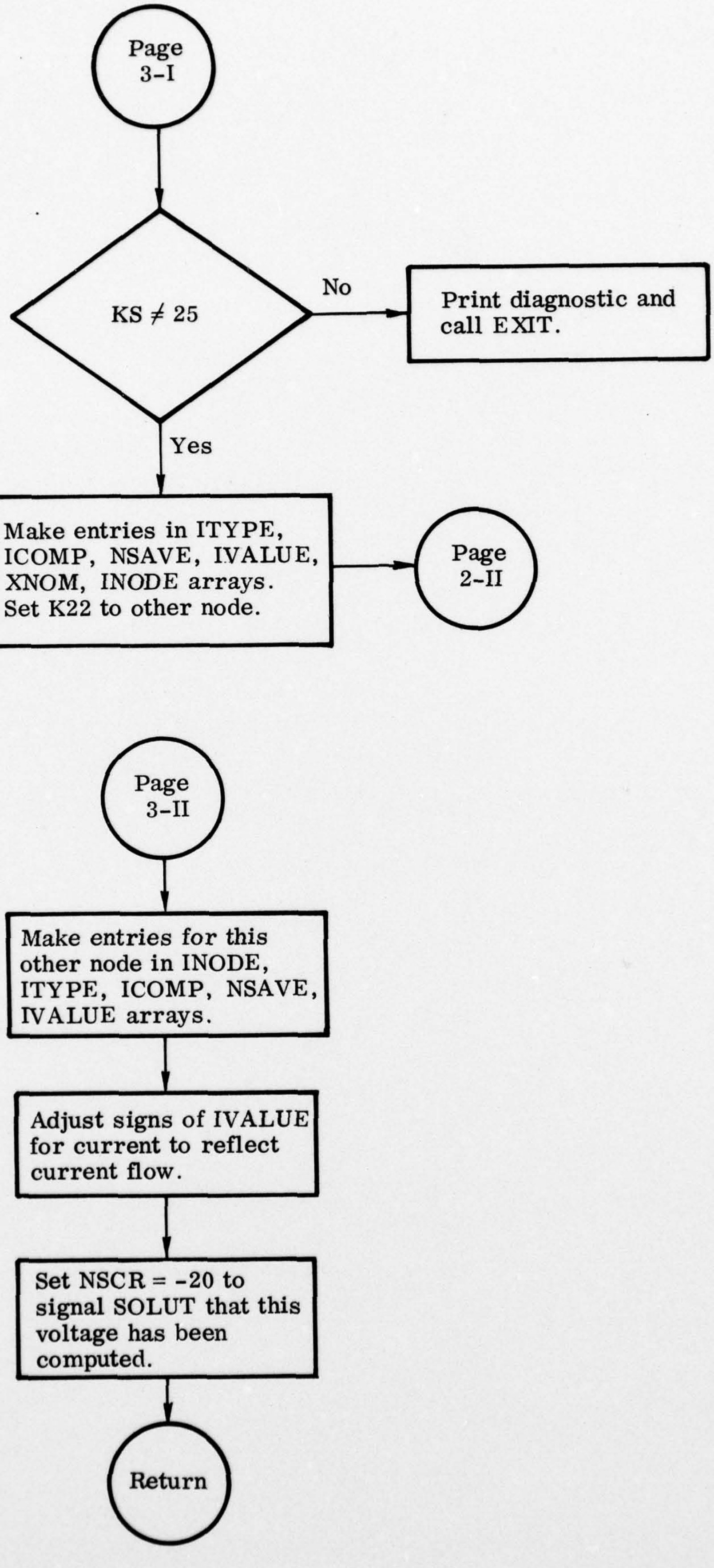
CURENT-1



Start
M = 0
DIAG = 0
C. E. = Circuit Element
KS ≠ 0
--i. e. have we already processed this equivalent circuit
No
Yes
Find NSAVE entry to correspond to this voltage C. E.
Find two entries in IVALUE array, compute current from resistor in NSAVE.
Return
Save value, type, and C. E. number of this voltage.
Did we find a node this C. E. comes from
No
DIAG = 1
Return
Yes
Page 2-I

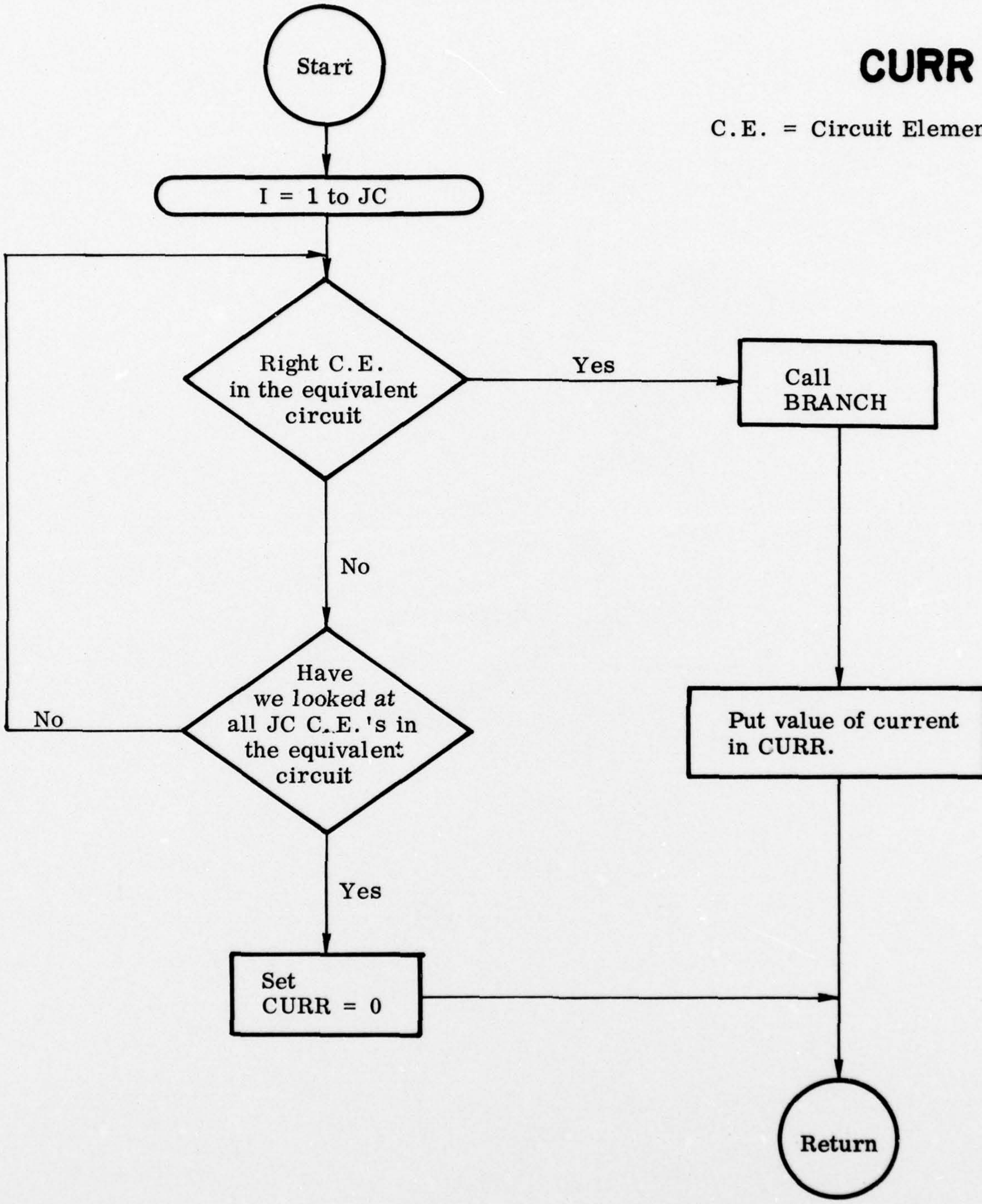
CURENT - 2


Page 2-I
Find NPOW entry for this voltage. Compute resistor and current.
Page 2-II
Is node we are looking at (to add resistor to) the last one of the equivalent circuit
Yes
Increase JC by 1; compute NB, the position in IVALUE array where the resistor entry will go; increase the link count of that node by one.
No
Move INODE, ITYPE, ICOMP, IVALUE arrays down to make room for new resistor.
Also move INODE arrays and increment the link counter array. JC = JC + 1.
Have we done this for both nodes that the voltage lies between
No
Page 3-I
Yes
Page 3-II

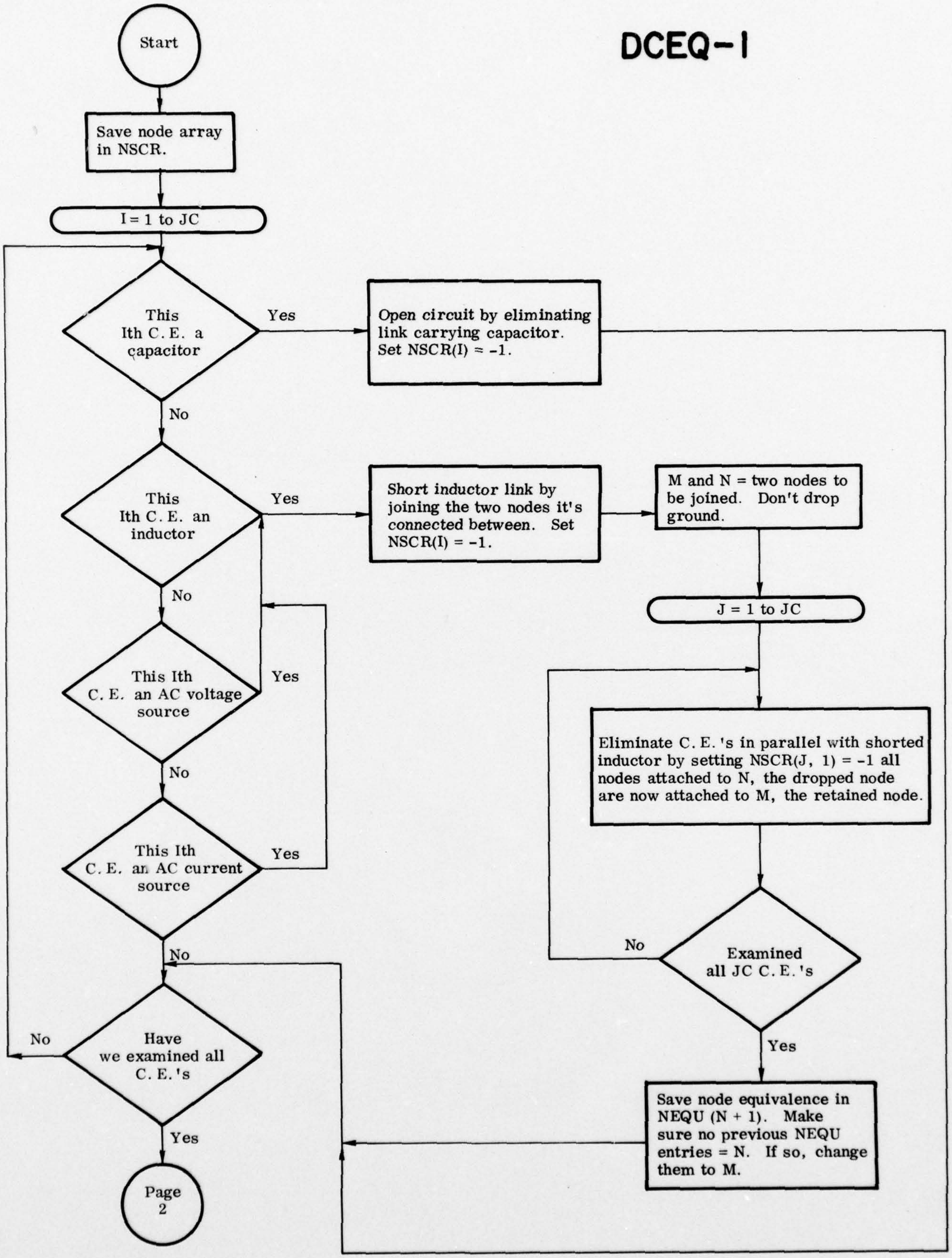

Page
3-I
KS ≠ 25
No
Print diagnostic and call EXIT.
Yes
Make entries in ITYPE, ICOMP, NSAVE, IVALUE, XNOM, INODE arrays. Set K22 to other node.
Page
2-II
Page
3-II
Make entries for this other node in INODE, ITYPE, ICOMP, NSAVE, IVALUE arrays.
Adjust signs of IVALUE for current to reflect current flow.
Set NSCR = -20 to signal SOLUT that this voltage has been computed.
Return

CURR

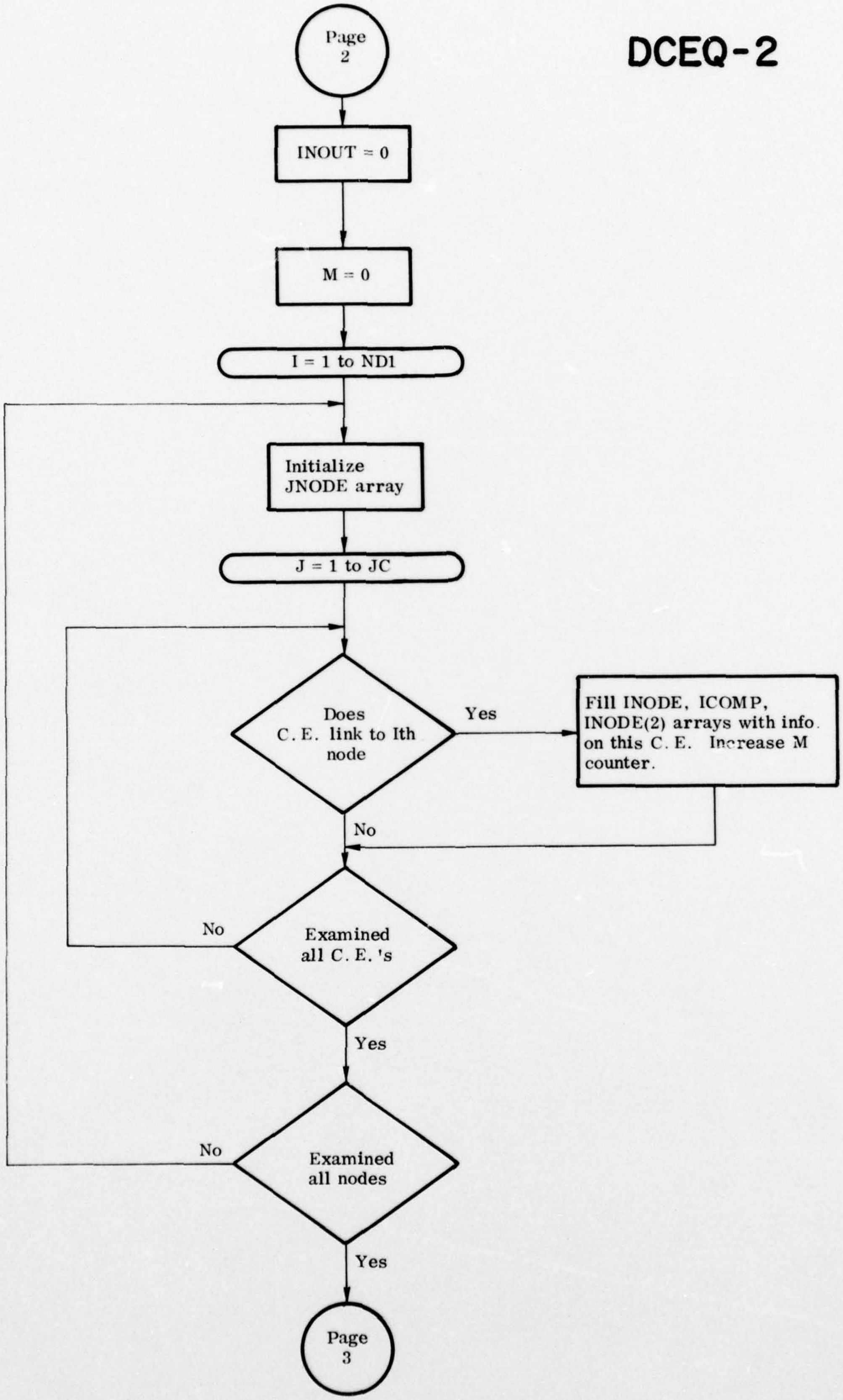
C.E. = Circuit Element



DCEQ-1

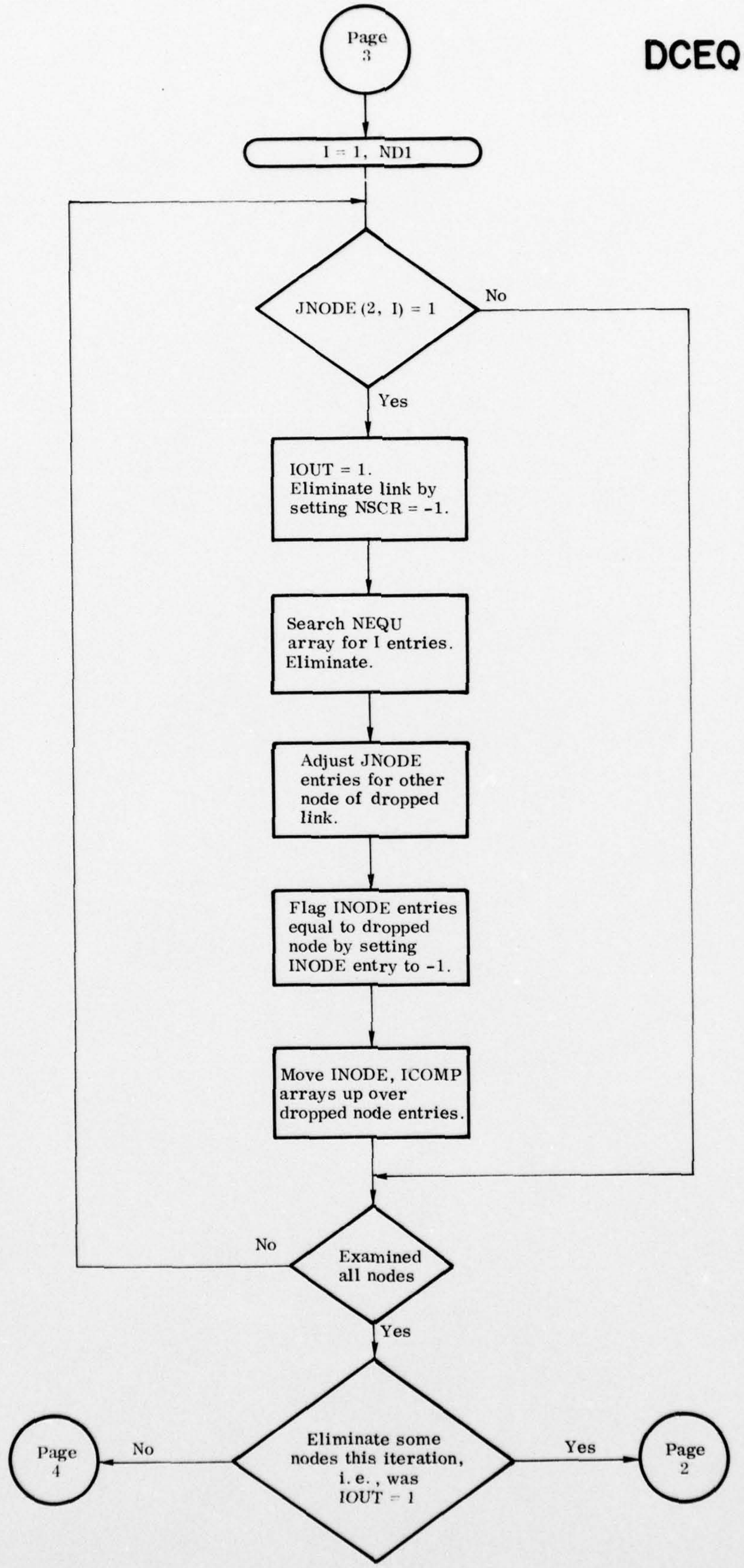

Start
Save node array in NSCR.
I = 1 to JC
This Ith C.E. a capacitor
Yes
Open circuit by eliminating link carrying capacitor. Set NSCR(I) = -1.
No
This Ith C.E. an inductor
Yes
Short inductor link by joining the two nodes it's connected between. Set NSCR(I) = -1.
M and N = two nodes to be joined. Don't drop ground.
J = 1 to JC
No
This Ith C.E. an AC voltage source
Yes
Eliminate C.E.'s in parallel with shorted inductor by setting NSCR(J, 1) = -1 all nodes attached to N, the dropped node are now attached to M, the retained node.
No
This Ith C.E. an AC current source
Yes
No
Examined all JC C.E.'s
No
Yes
Have we examined all C.E.'s
No
Save node equivalence in NEQU (N + 1). Make sure no previous NEQU entries = N. If so, change them to M.
Yes
Page 2

DCEQ-2



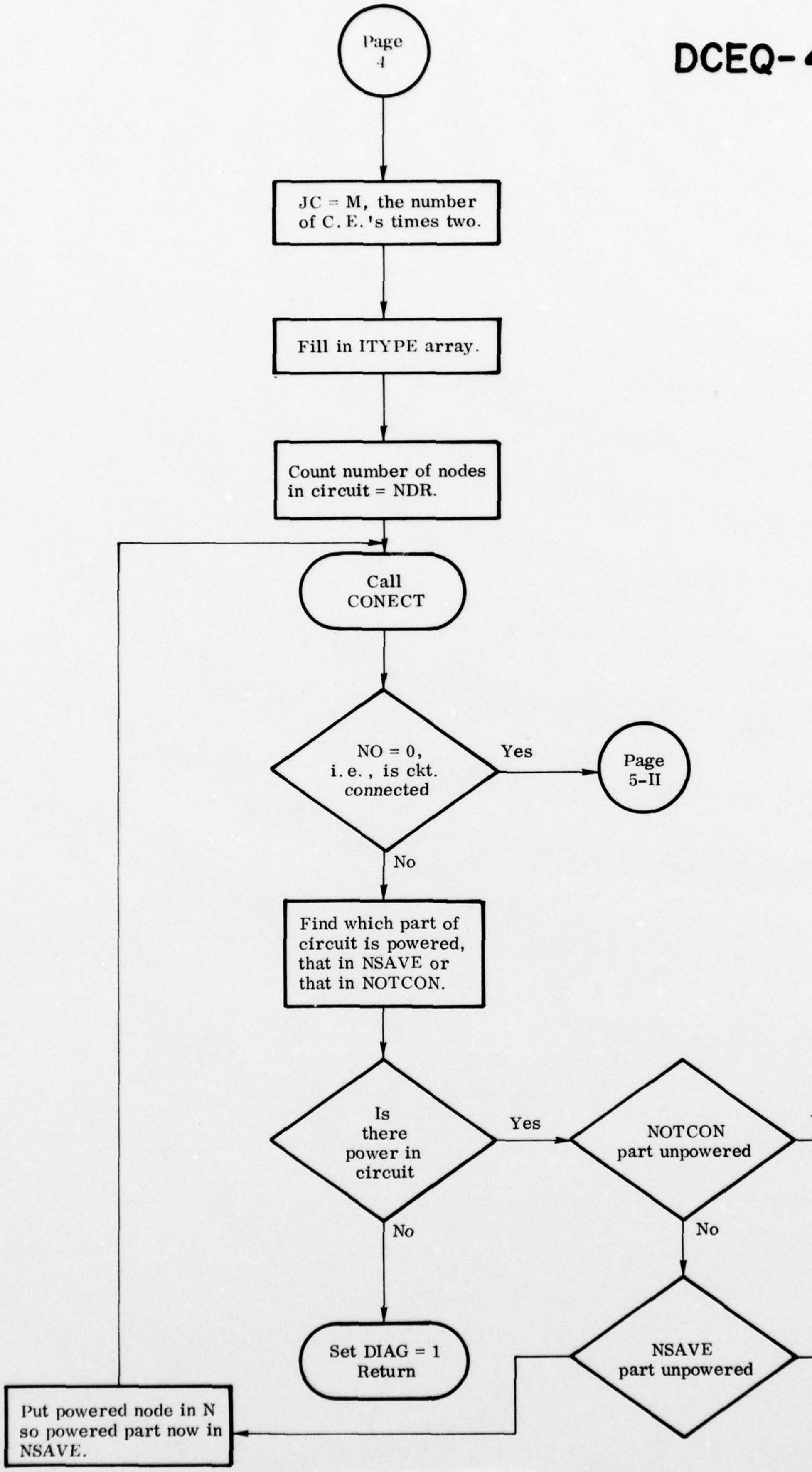
Page 2
INOUT = 0
M = 0
I = 1 to ND1
Initialize JNODE array
J = 1 to JC
Does C. E. link to Ith node
Yes
Fill INODE, ICOMP, INODE(2) arrays with info. on this C. E. Increase M counter.
No
Examined all C. E. 's
No
Yes
Examined all nodes
No
Yes
Page 3

DCEQ-3

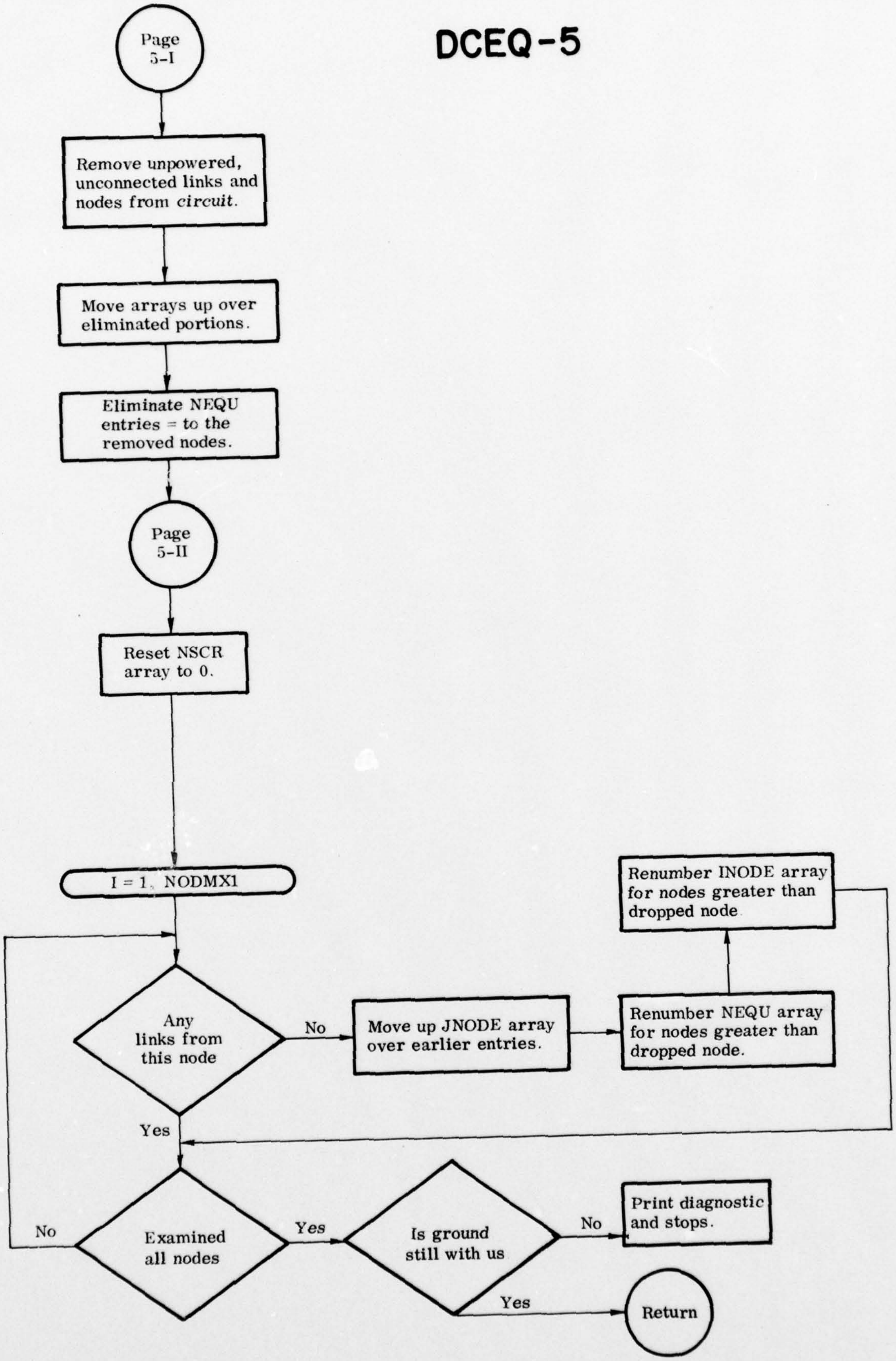


Page 3
I = 1, ND1
JNODE (2, I) = 1
No
Yes
IOUT = 1.
Eliminate link by setting NSCR = -1.
Search NEQU array for I entries. Eliminate.
Adjust JNODE entries for other node of dropped link.
Flag INODE entries equal to dropped node by setting INODE entry to -1.
Move INODE, ICOMP arrays up over dropped node entries.
No
Examined all nodes
Yes
Page 4
No
Eliminate some nodes this iteration, i.e., was IOUT = 1
Yes
Page 2

DCEQ-4

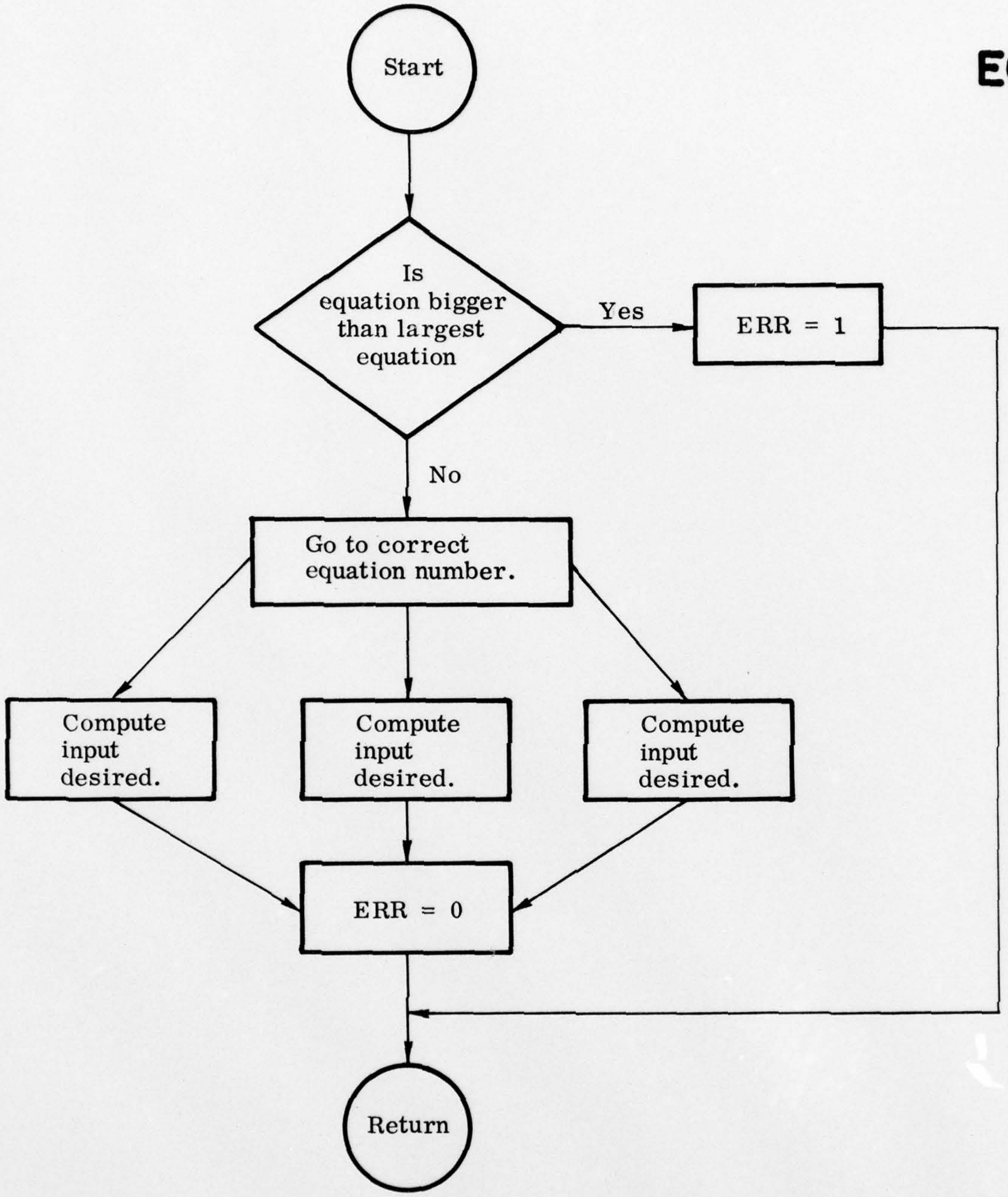

Page 4
JC = M, the number of C.E.'s times two.
Fill in ITYPE array.
Count number of nodes in circuit = NDR.
Call CONECT
NO = 0, i.e., is ckt. connected
Yes
Page 5-II
No
Find which part of circuit is powered, that in NSAVE or that in NOTCON.
Is there power in circuit
Yes
NOTCON part unpowered
Yes
Page 5-I
No
No
Set DIAG = 1 Return
NSAVE part unpowered
Page 5-II
Put powered node in N so powered part now in NSAVE.

DCEQ-5

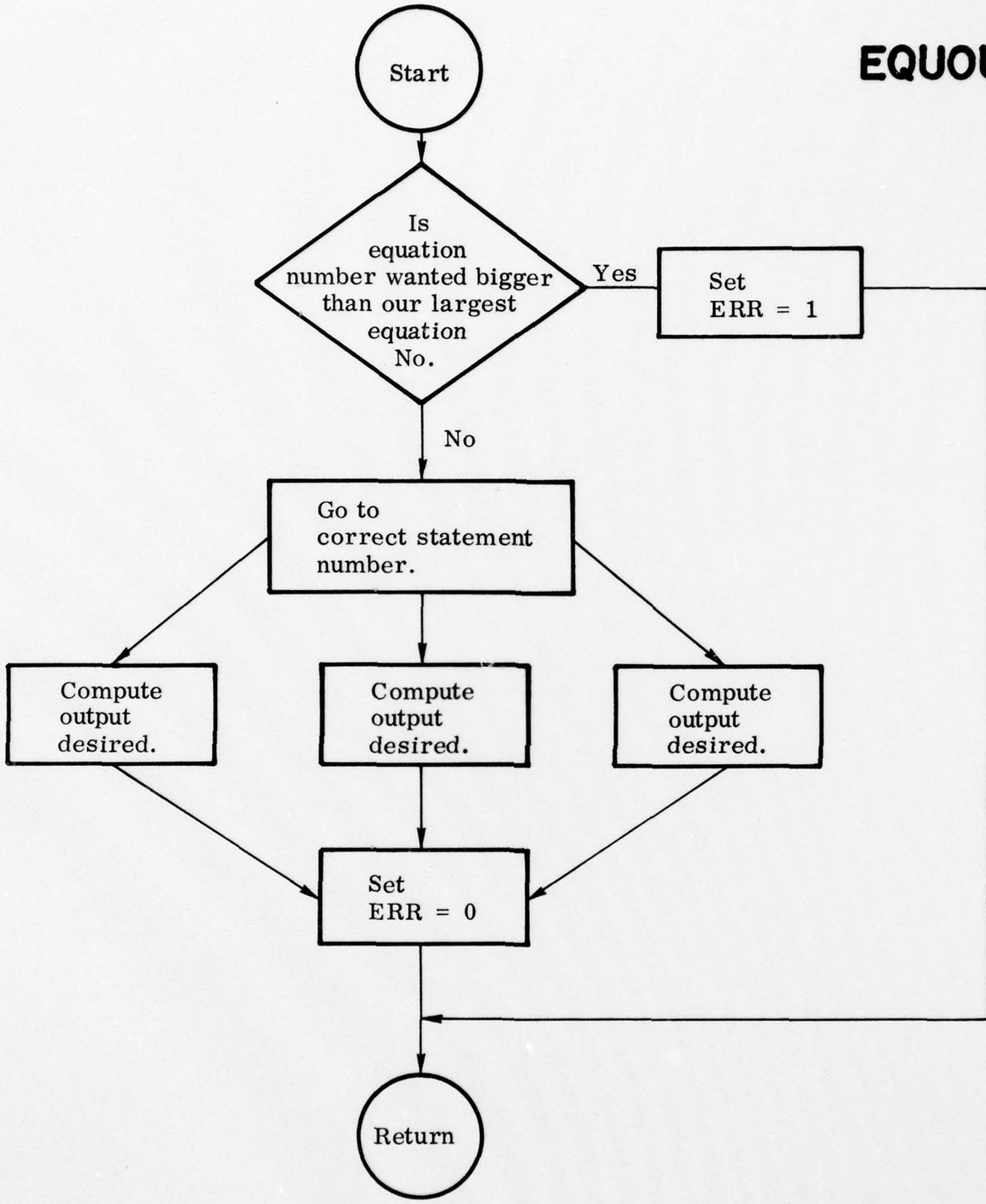


Page 5-I
Remove unpowered, unconnected links and nodes from *circuit*.
Move arrays up over eliminated portions.
Eliminate NEQU entries = to the removed nodes.
Page 5-II
Reset NSCR array to 0.
I = 1, NODMX1
Any links from this node
No
Move up JNODE array over earlier entries.
Renumber NEQU array for nodes greater than dropped node.
Renumber INODE array for nodes greater than dropped node.
Yes
No
Examined all nodes
Yes
Is ground still with us
No
Print diagnostic and stops.
Yes
Return

EQUIN

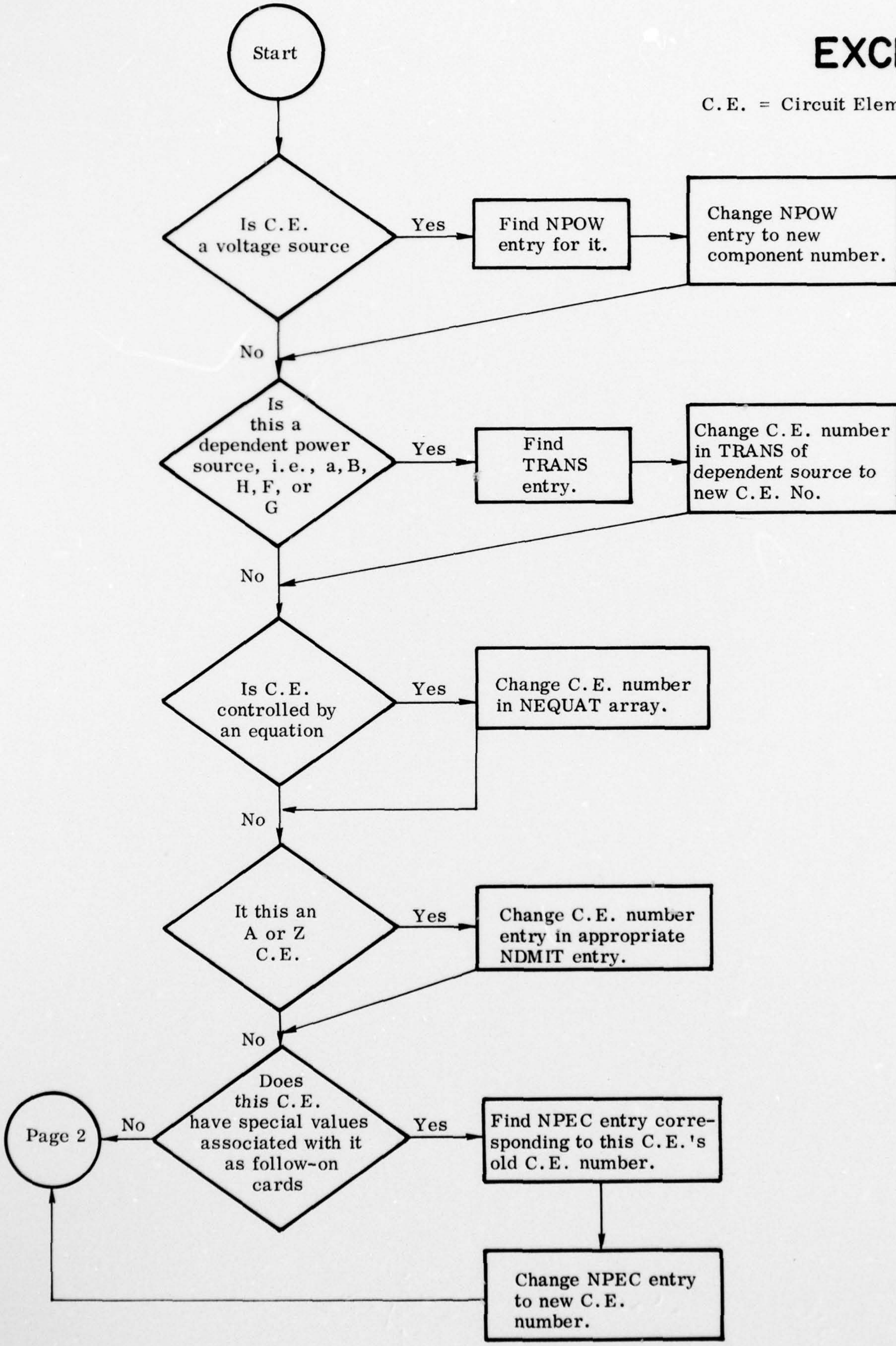

Start
Is equation bigger than largest equation
Yes
ERR = 1
No
Go to correct equation number.
Compute input desired.
Compute input desired.
Compute input desired.
ERR = 0
Return

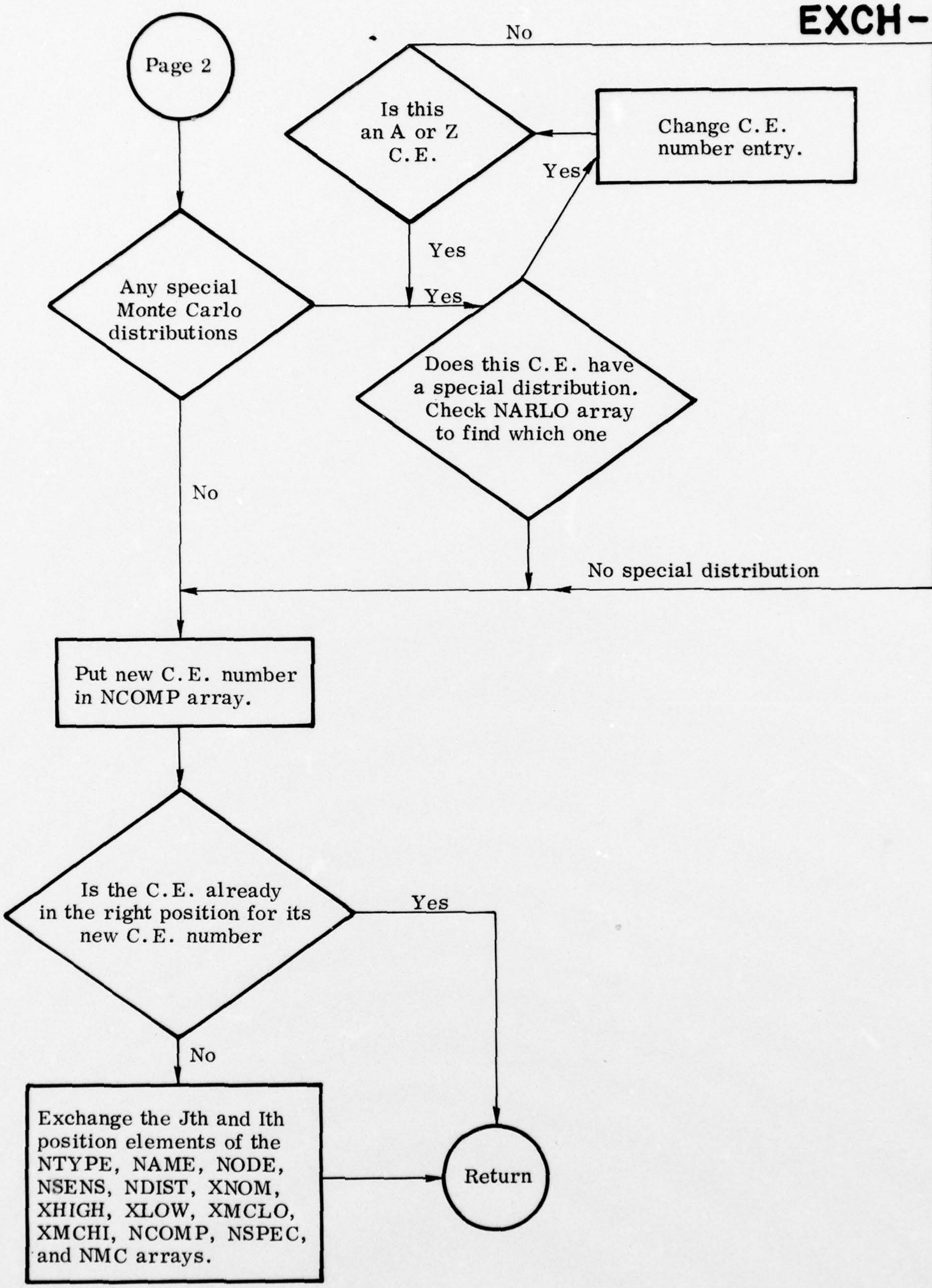
EQUOUT


Start
Is equation number wanted bigger than our largest equation No.
Yes
Set ERR = 1
No
Go to correct statement number.
Compute output desired.
Compute output desired.
Compute output desired.
Set ERR = 0
Return

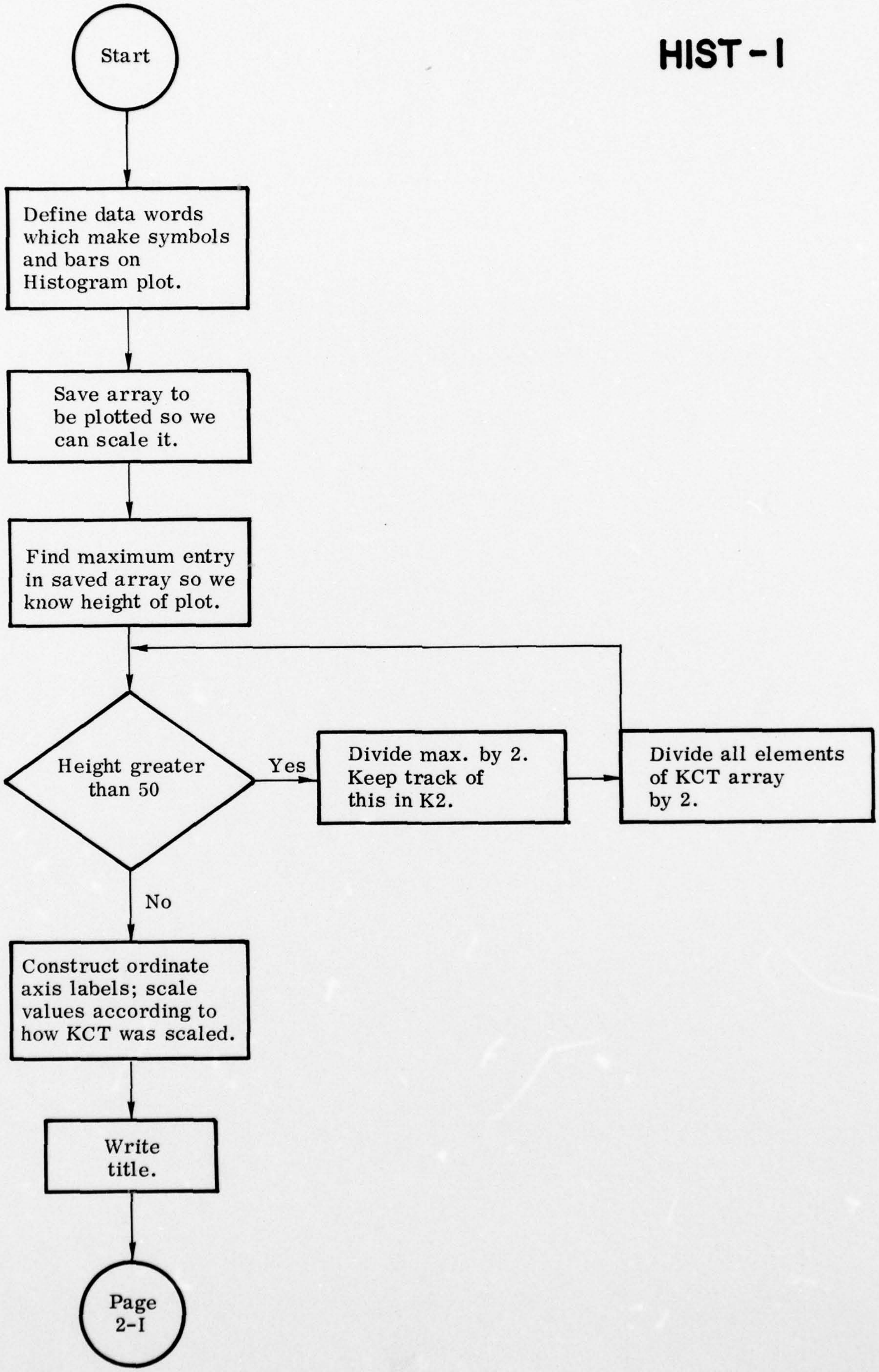
EXCH-1

C.E. = Circuit Element

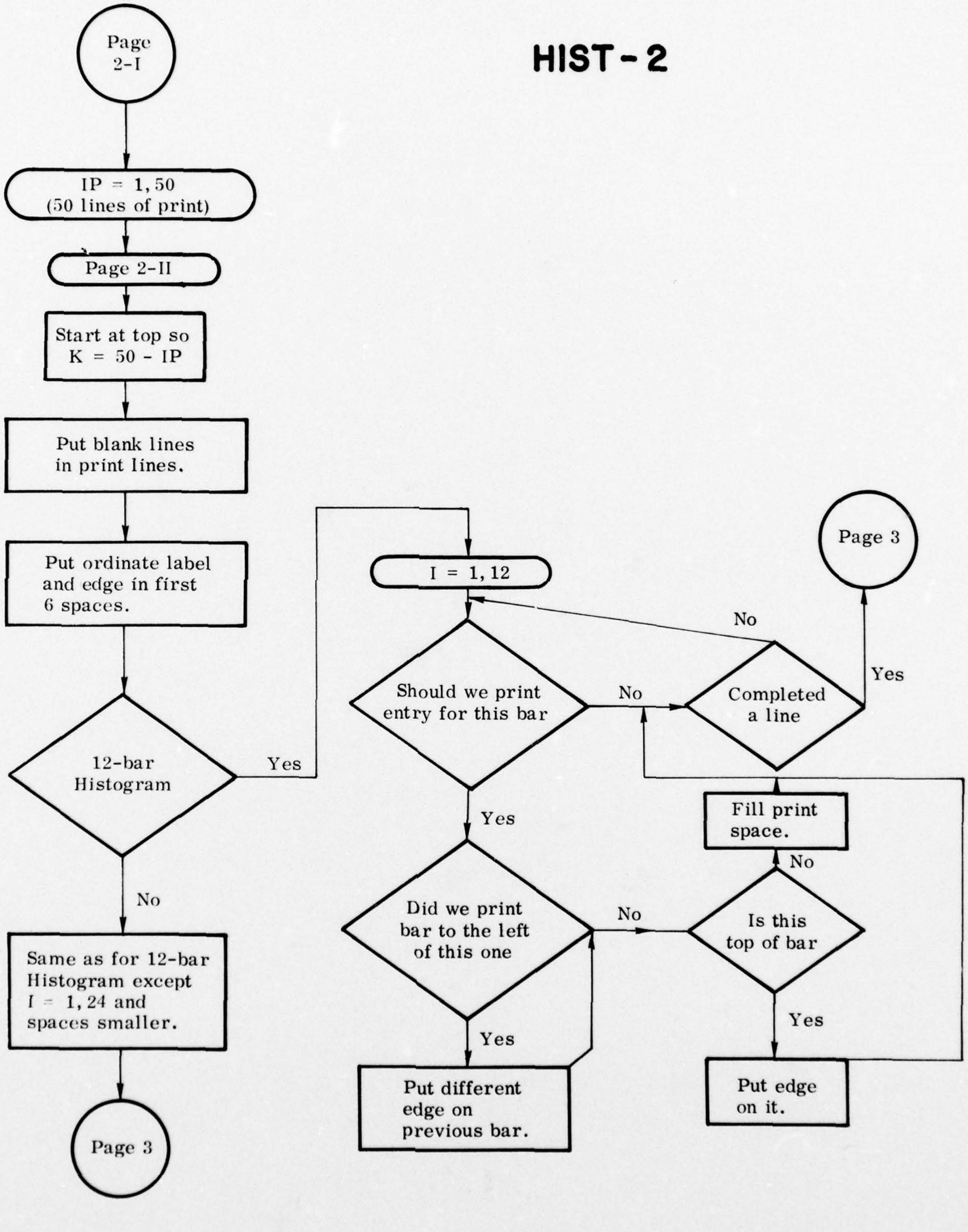



Page 2
Any special Monte Carlo distributions
Yes
No
Is this an A or Z C.E.
No
Yes
Change C.E. number entry.
Yes
Does this C.E. have a special distribution. Check NARLO array to find which one
Yes
No special distribution
Put new C.E. number in NCOMP array.
Is the C.E. already in the right position for its new C.E. number
Yes
No
Exchange the Jth and Ith position elements of the NTYPE, NAME, NODE, NSENS, NDIST, XNOM, XHIGH, XLOW, XMCLO, XMCHI, NCOMP, NSPEC, and NMC arrays.
Return

HIST-1

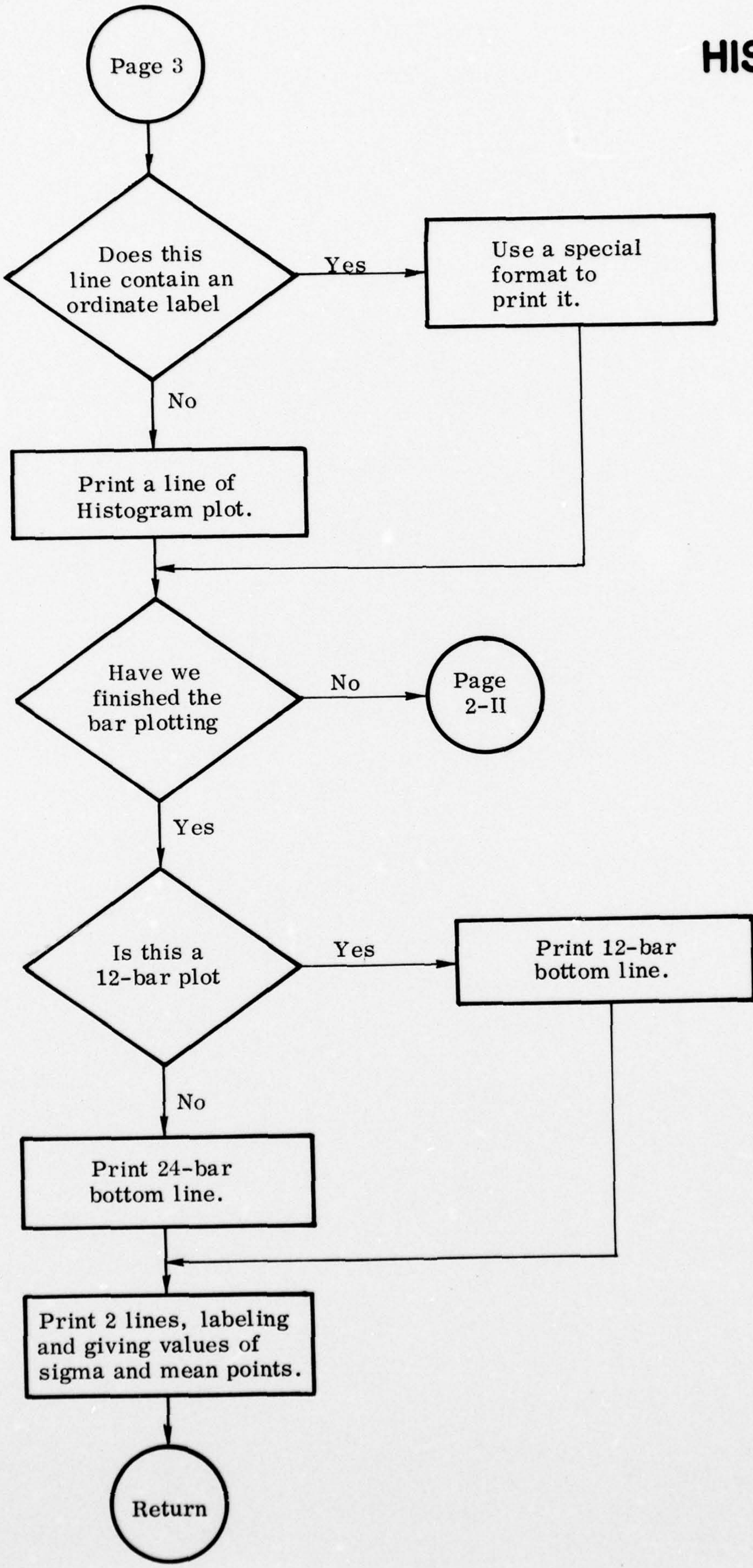

Start
Define data words which make symbols and bars on Histogram plot.
Save array to be plotted so we can scale it.
Find maximum entry in saved array so we know height of plot.
Height greater than 50
Yes
Divide max. by 2. Keep track of this in K2.
Divide all elements of KCT array by 2.
No
Construct ordinate axis labels; scale values according to how KCT was scaled.
Write title.
Page 2-I

HIST-2

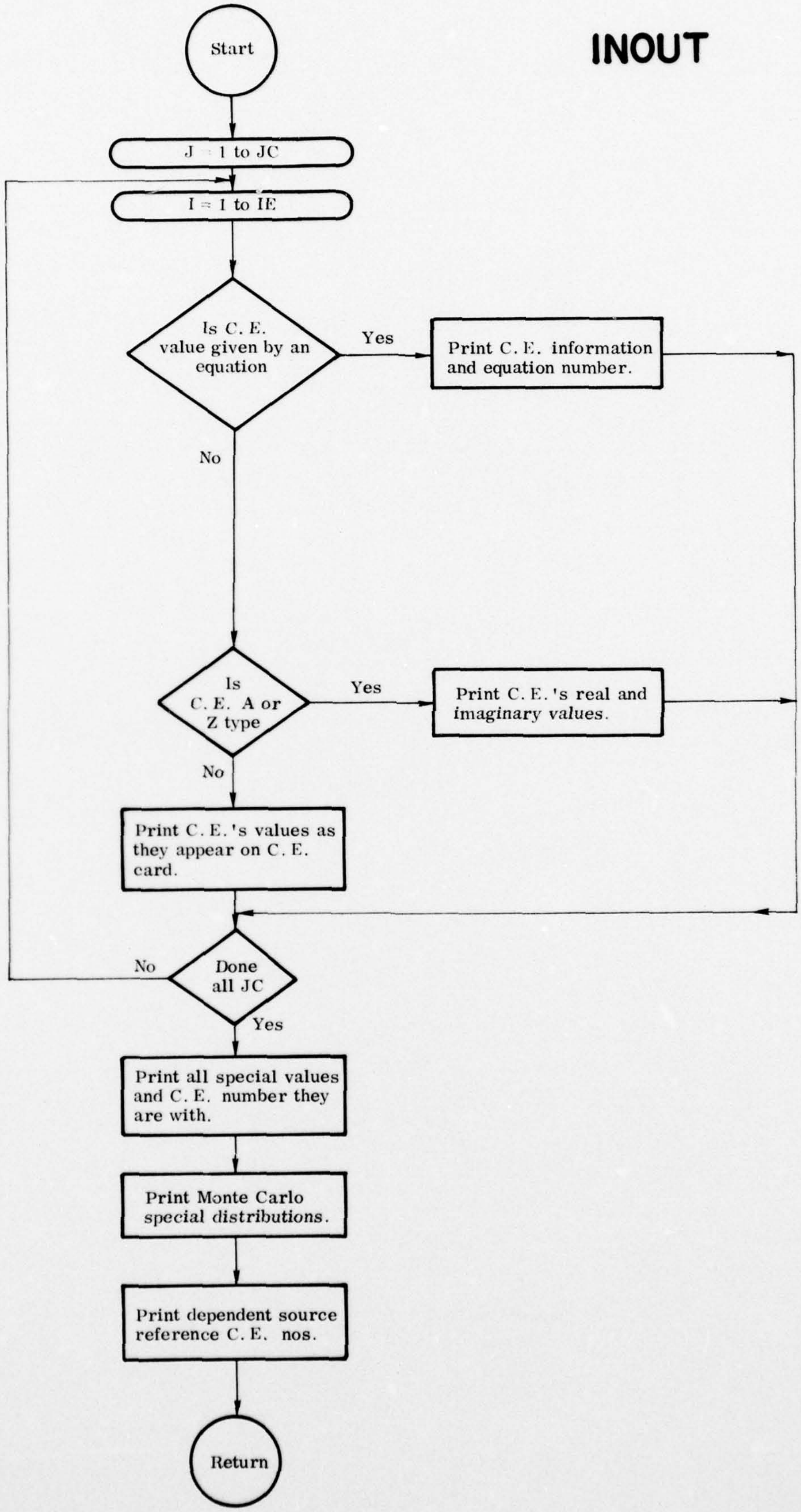


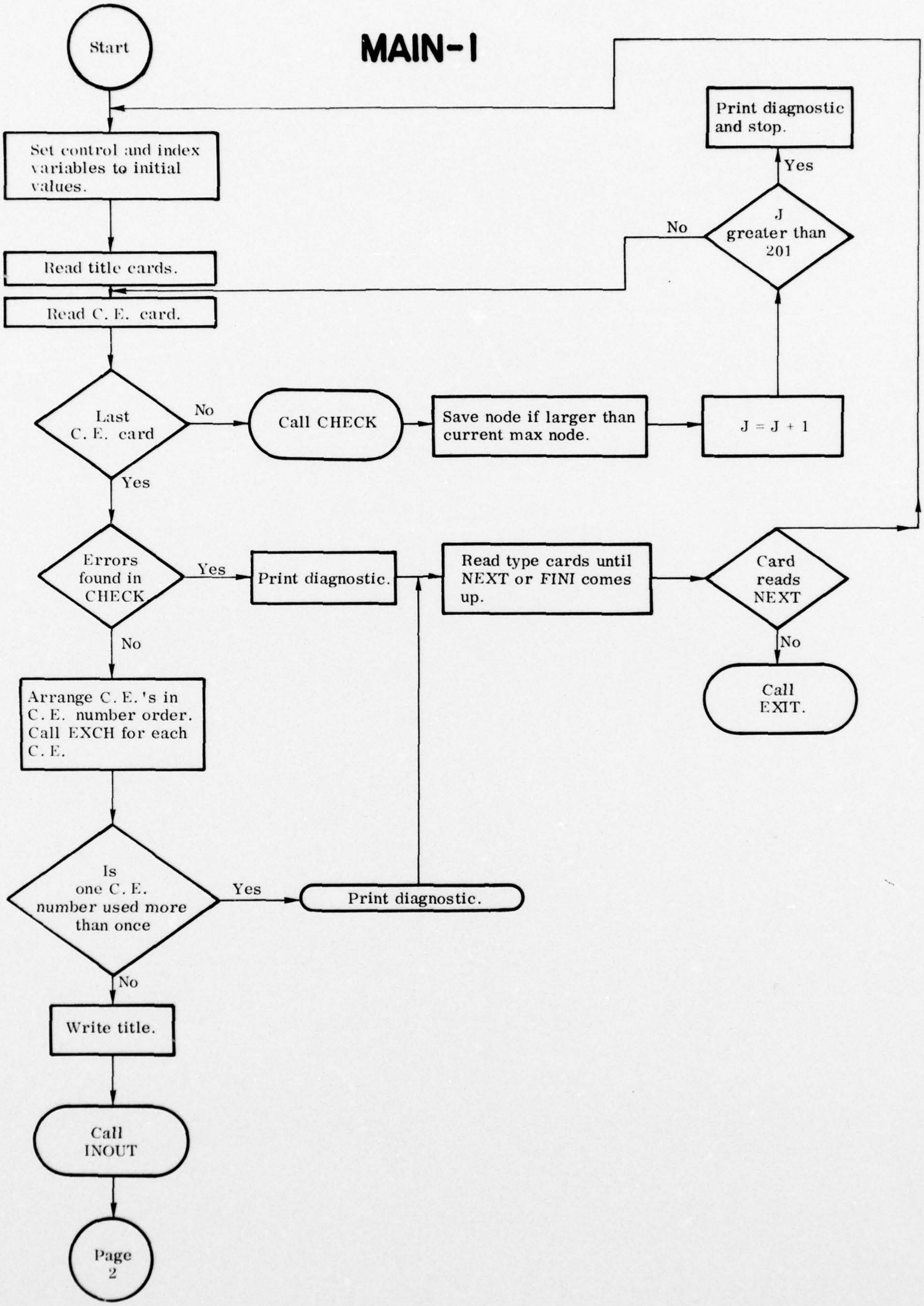
Page 2-I
IP = 1,50 (50 lines of print)
Page 2-II
Start at top so K = 50 - IP
Put blank lines in print lines.
Put ordinate label and edge in first 6 spaces.
12-bar Histogram
Yes
No
Same as for 12-bar Histogram except I = 1, 24 and spaces smaller.
Page 3
I = 1, 12
Should we print entry for this bar
No
Yes
Completed a line
No
Yes
Page 3
Did we print bar to the left of this one
No
Yes
Put different edge on previous bar.
Is this top of bar
No
Yes
Fill print space.
Put edge on it.

HIST - 3

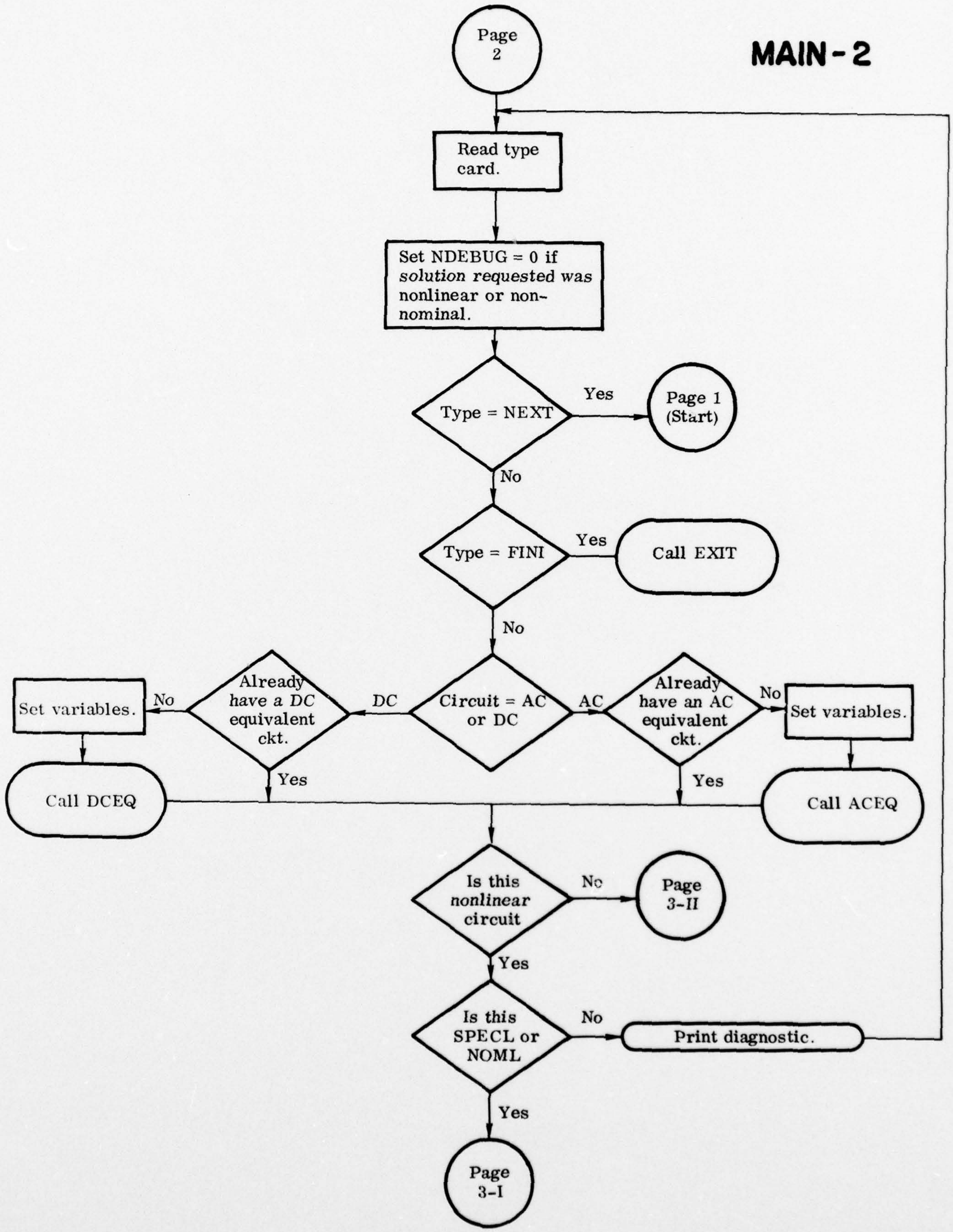

Page 3
Does this line contain an ordinate label
Yes
Use a special format to print it.
No
Print a line of Histogram plot.
Have we finished the bar plotting
No
Page 2-II
Yes
Is this a 12-bar plot
Yes
Print 12-bar bottom line.
No
Print 24-bar bottom line.
Print 2 lines, labeling and giving values of sigma and mean points.
Return

INOUT

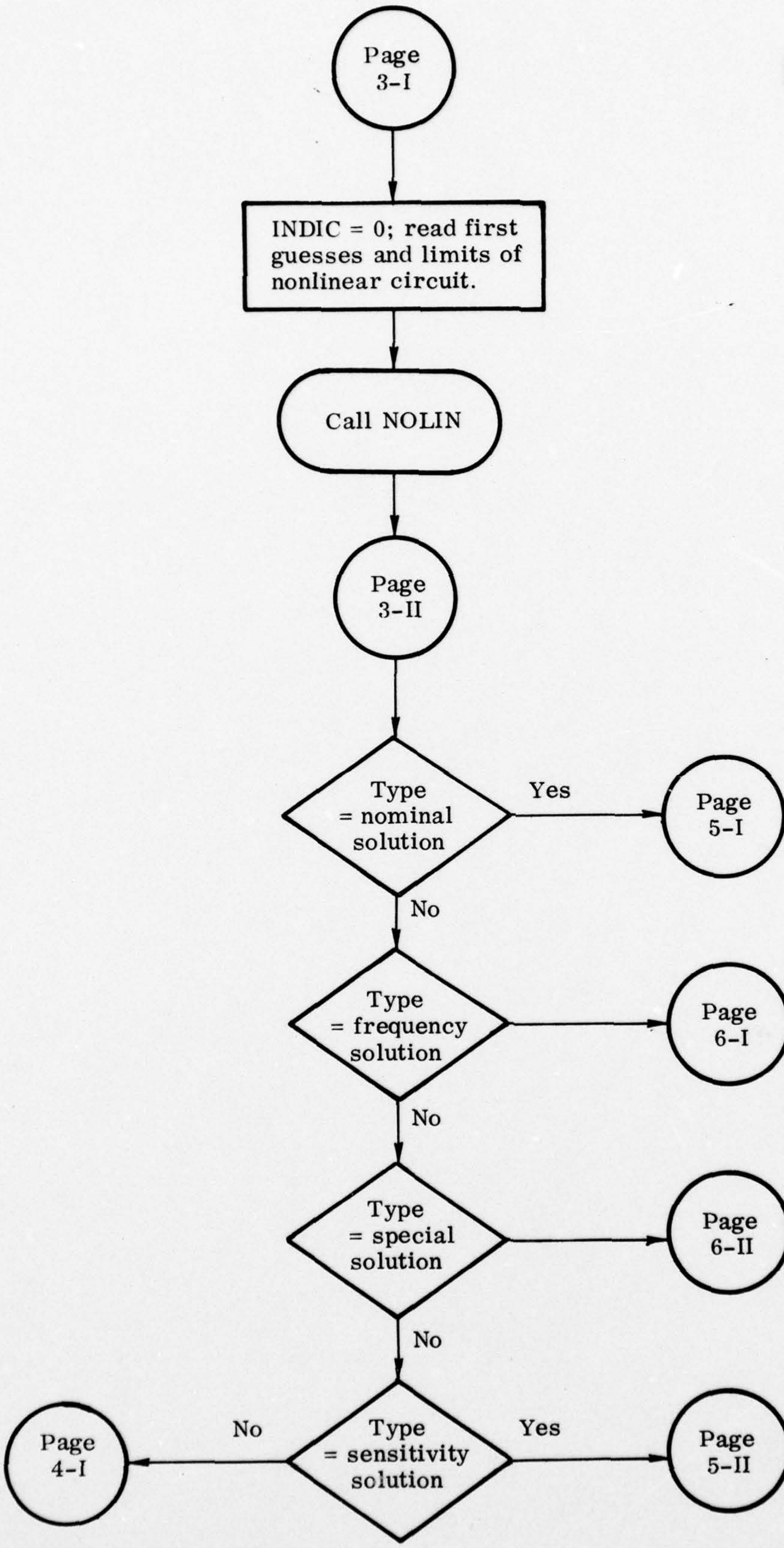

Start
J = 1 to JC
I = 1 to IE
Is C. E. value given by an equation
Yes
Print C. E. information and equation number.
No
Is C. E. A or Z type
Yes
Print C. E.'s real and imaginary values.
No
Print C. E.'s values as they appear on C. E. card.
No
Done all JC
Yes
Print all special values and C. E. number they are with.
Print Monte Carlo special distributions.
Print dependent source reference C. E. nos.
Return


Start
MAIN-1
Set control and index variables to initial values.
Read title cards.
Read C. E. card.
Last C. E. card
No
Yes
Call CHECK
Save node if larger than current max node.
J = J + 1
J greater than 201
No
Yes
Print diagnostic and stop.
Errors found in CHECK
Yes
No
Print diagnostic.
Read type cards until NEXT or FINI comes up.
Card reads NEXT
No
Call EXIT.
Arrange C. E.'s in C. E. number order. Call EXCH for each C. E.
Is one C. E. number used more than once
Yes
No
Print diagnostic.
Write title.
Call INOUT
Page 2

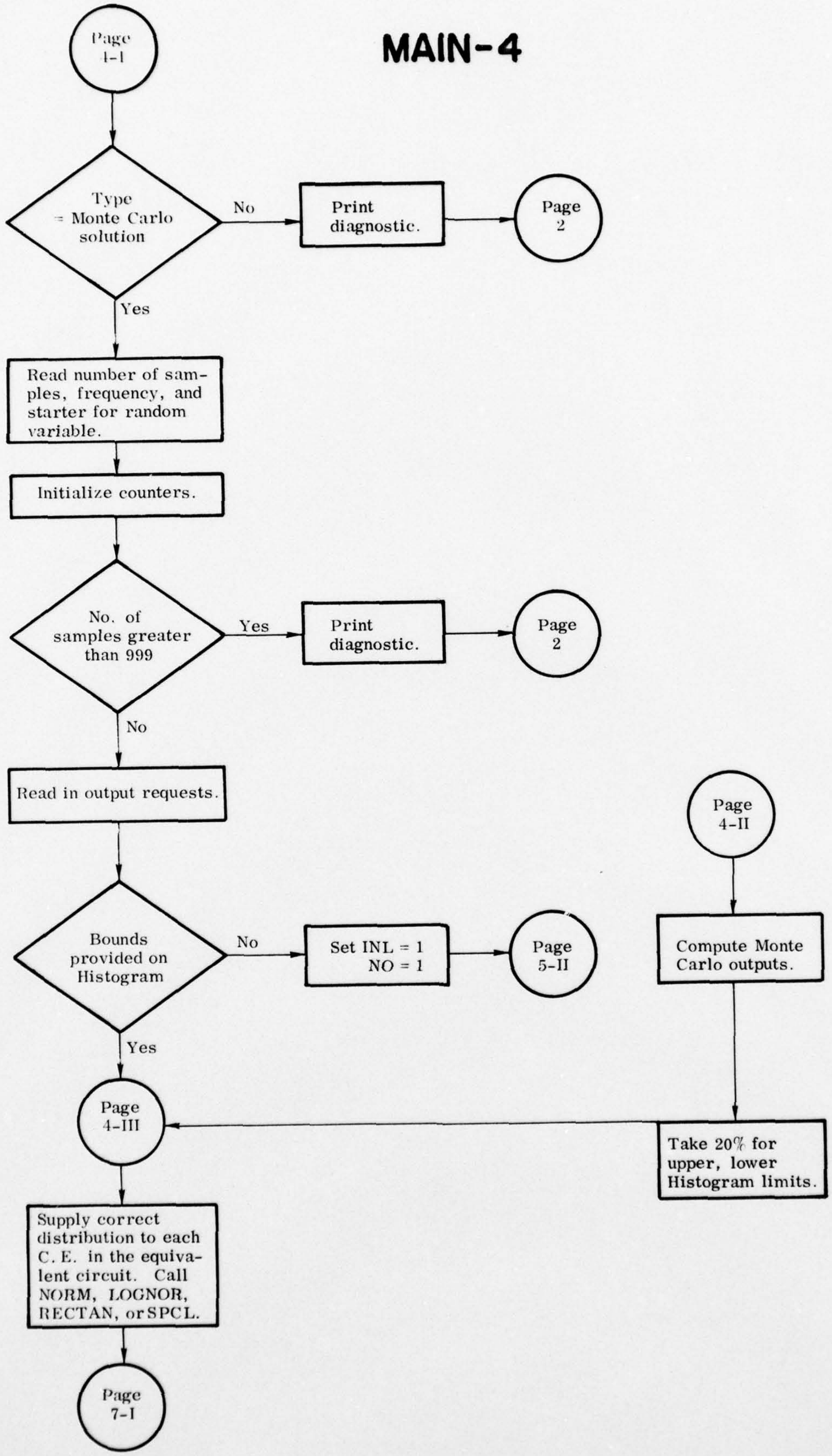
MAIN - 2


Page 2
Read type card.
Set NDEBUG = 0 if *solution requested* was nonlinear or non-nominal.
Type = NEXT
Yes
Page 1 (Start)
No
Type = FINI
Yes
Call EXIT
No
Circuit = AC or DC
DC
Already *have a DC* equivalent ckt.
No
Set variables.
Call DCEQ
Yes
AC
Already have an AC equivalent ckt.
No
Set variables.
Call ACEQ
Yes
Is this *nonlinear* circuit
No
Page 3-II
Yes
Is this SPECL or NOML
No
Print diagnostic.
Yes
Page 3-I

MAIN-3

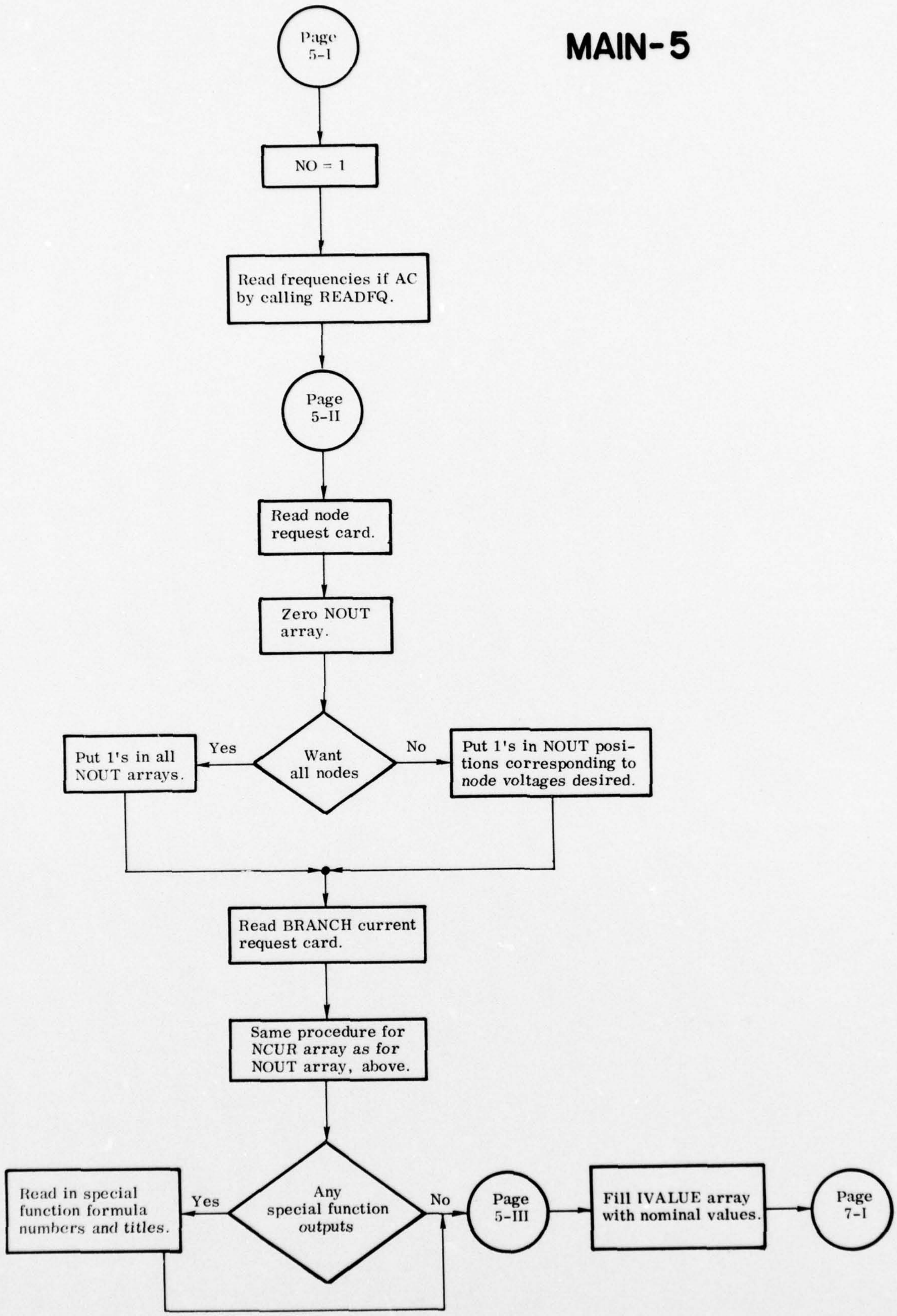

Page
3-I
INDIC = 0; read first
guesses and limits of
nonlinear circuit.
Call NOLIN
Page
3-II
Type
= nominal
solution
Yes
Page
5-I
No
Type
= frequency
solution
Page
6-I
No
Type
= special
solution
Page
6-II
No
Type
= sensitivity
solution
Yes
Page
5-II
No
Page
4-I

MAIN-4



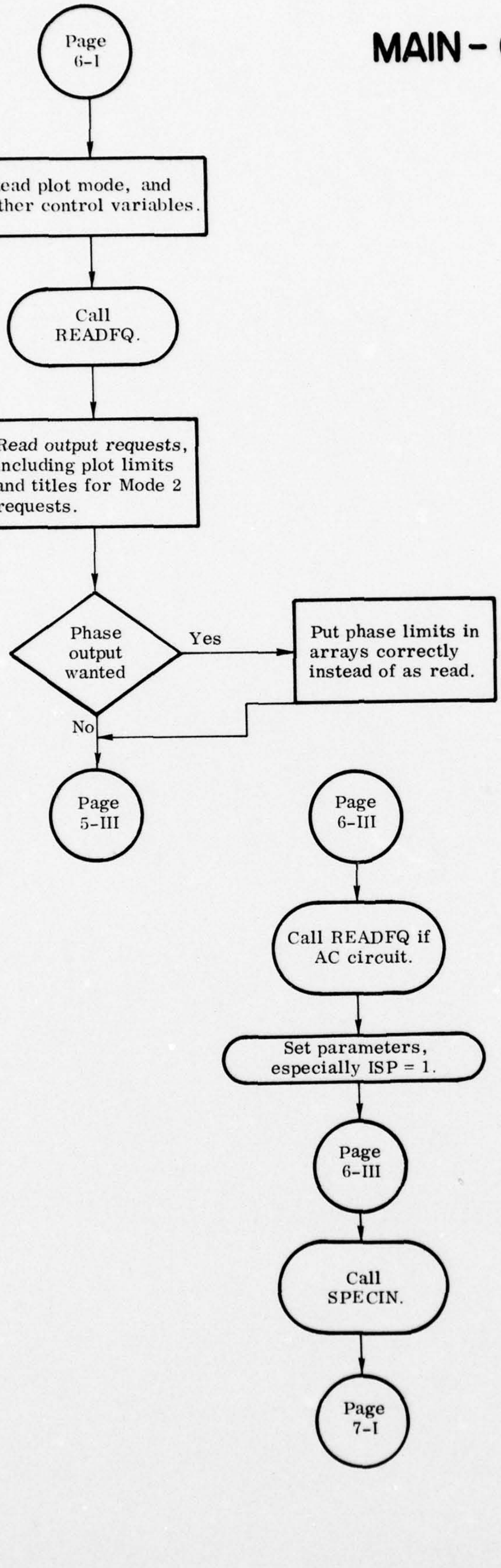
Page 4-I
Type = Monte Carlo solution
No
Print diagnostic.
Page 2
Yes
Read number of samples, frequency, and starter for random variable.
Initialize counters.
No. of samples greater than 999
Yes
Print diagnostic.
Page 2
No
Read in output requests.
Page 4-II
Bounds provided on Histogram
No
Set INL = 1
NO = 1
Page 5-II
Compute Monte Carlo outputs.
Yes
Page 4-III
Take 20% for upper, lower Histogram limits.
Supply correct distribution to each C. E. in the equivalent circuit. Call NORM, LOGNOR, RECTAN, or SPCL.
Page 7-I

MAIN-5



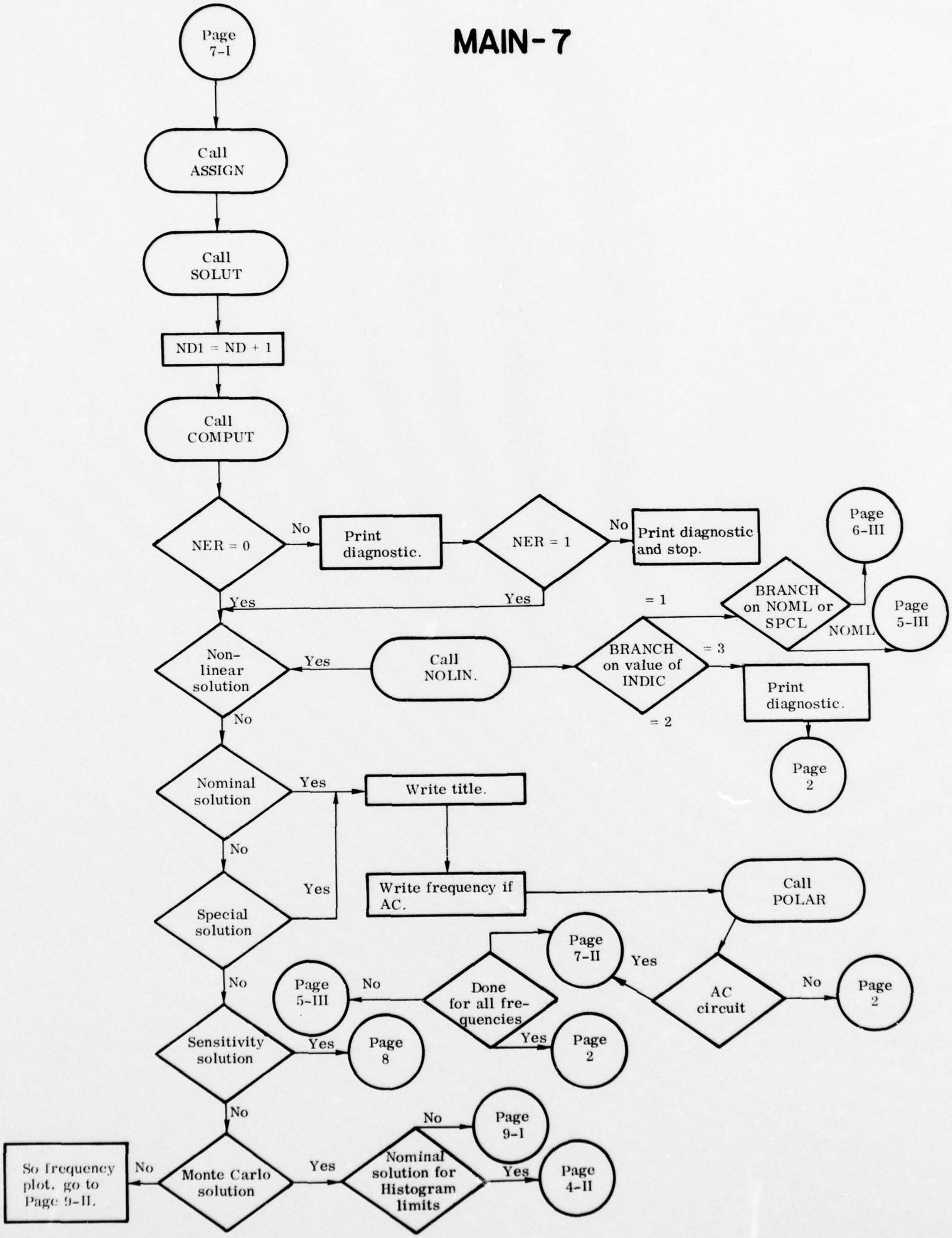
Page 5-I
NO = 1
Read frequencies if AC by calling READFQ.
Page 5-II
Read node request card.
Zero NOUT array.
Want all nodes
Yes
Put 1's in all NOUT arrays.
No
Put 1's in NOUT positions corresponding to node voltages desired.
Read BRANCH current request card.
Same procedure for NCUR array as for NOUT array, above.
Any special function outputs
Yes
Read in special function formula numbers and titles.
No
Page 5-III
Fill IVALUE array with nominal values.
Page 7-I

MAIN - 6

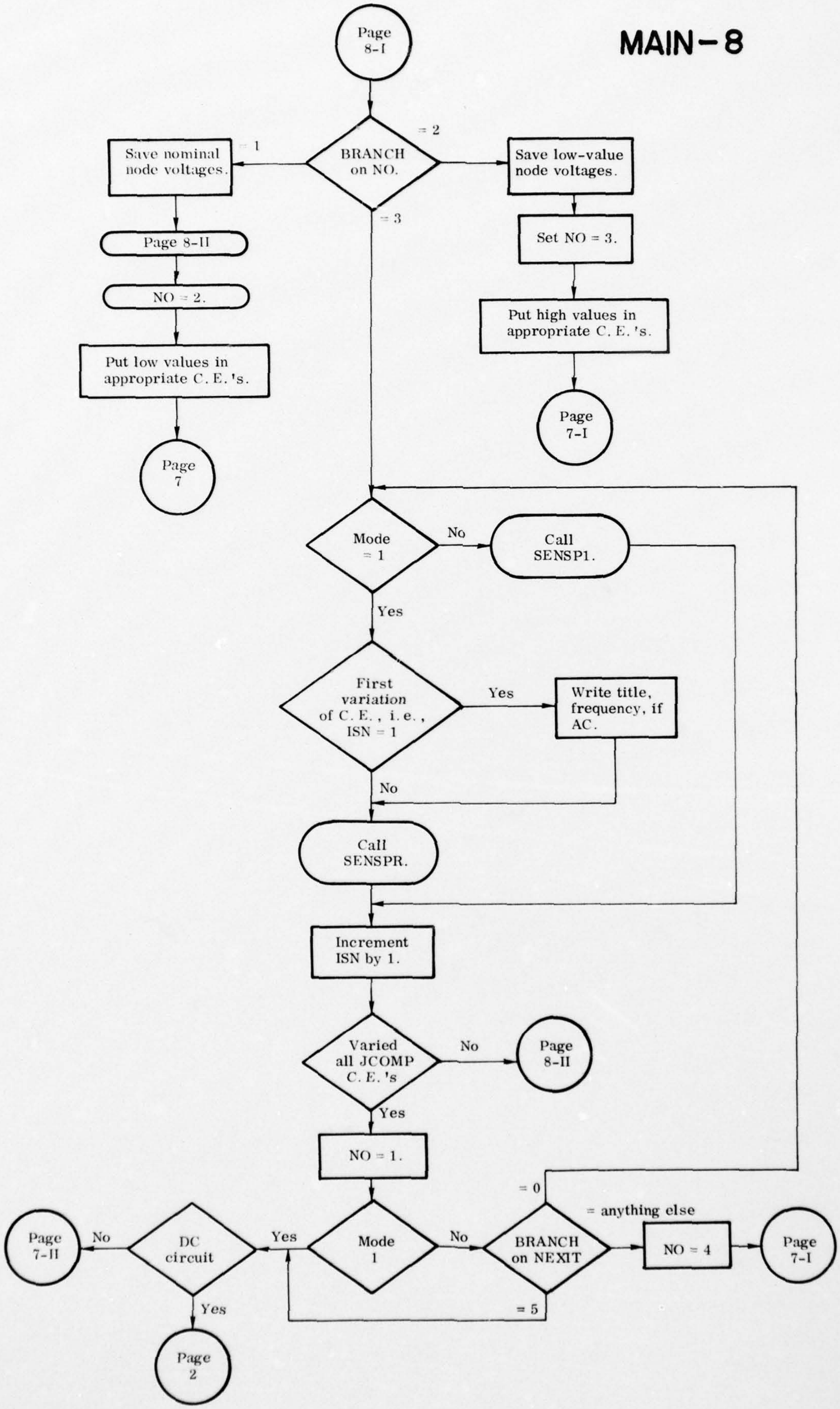


Page 6-I
Read plot mode, and other control variables.
Call READFQ.
Read output requests, including plot limits and titles for Mode 2 requests.
Phase output wanted
Yes
No
Put phase limits in arrays correctly instead of as read.
Page 5-III
Page 6-III
Call READFQ if AC circuit.
Set parameters, especially ISP = 1.
Page 6-III
Call SPECIN.
Page 7-I

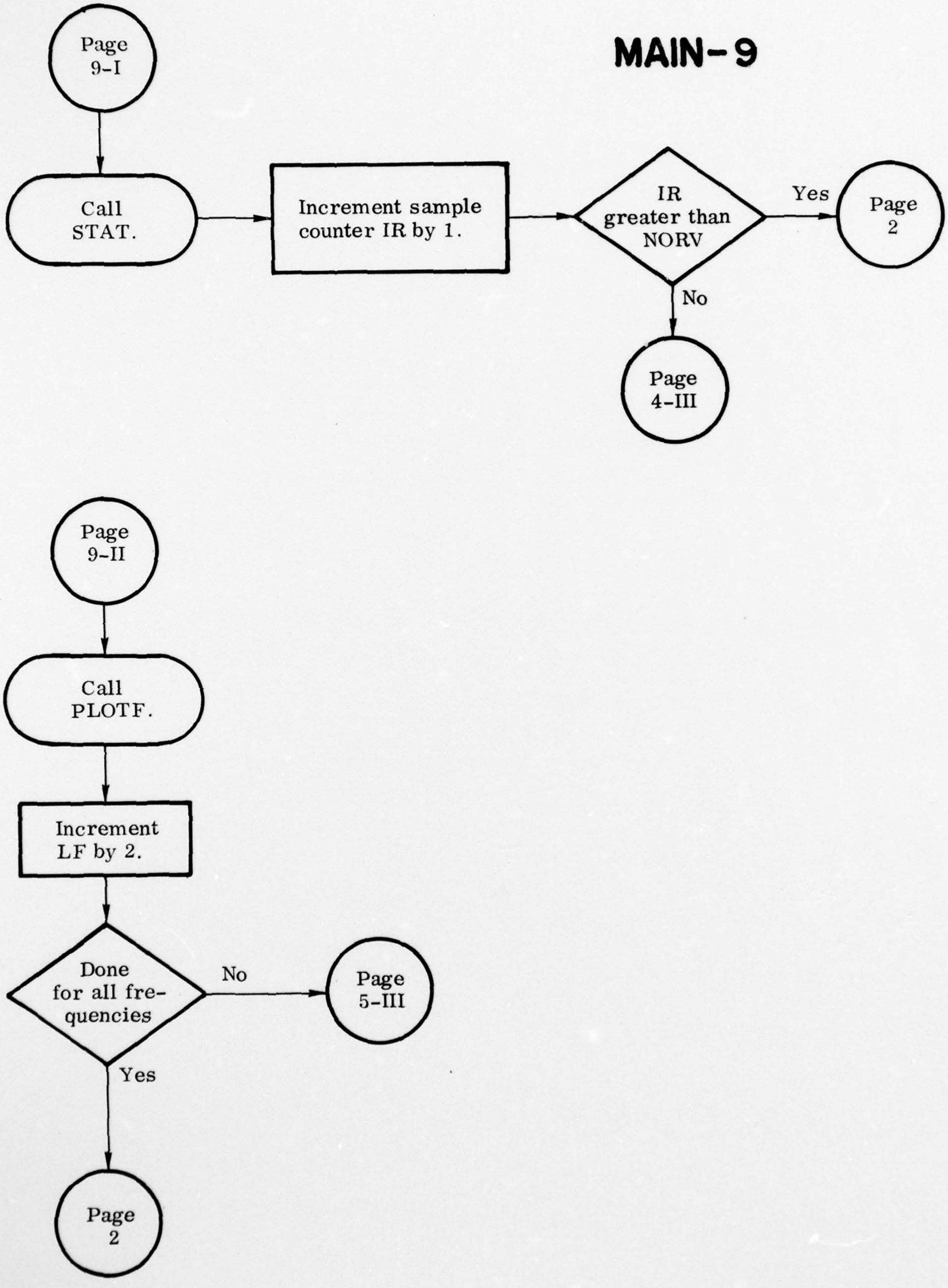
MAIN-7



Page 7-I
Call ASSIGN
Call SOLUT
ND1 = ND + 1
Call COMPUT
NER = 0
No
Print diagnostic.
NER = 1
No
Print diagnostic and stop.
Yes
Yes
Page 6-III
BRANCH on NOML or SPCL
= 1
Page 5-III
NOML
Non-linear solution
Yes
Call NOLIN.
BRANCH on value of INDIC
= 3
Print diagnostic.
= 2
No
Page 2
Nominal solution
Yes
Write title.
No
Yes
Write frequency if AC.
Call POLAR
Special solution
Page 7-II
Yes
No
Page 5-III
No
Done for all frequencies
AC circuit
No
Page 2
Sensitivity solution
Yes
Page 8
Yes
Page 2
No
No
Page 9-I
So frequency plot, go to Page 9-II.
No
Monte Carlo solution
Yes
Nominal solution for Histogram limits
Yes
Page 4-II

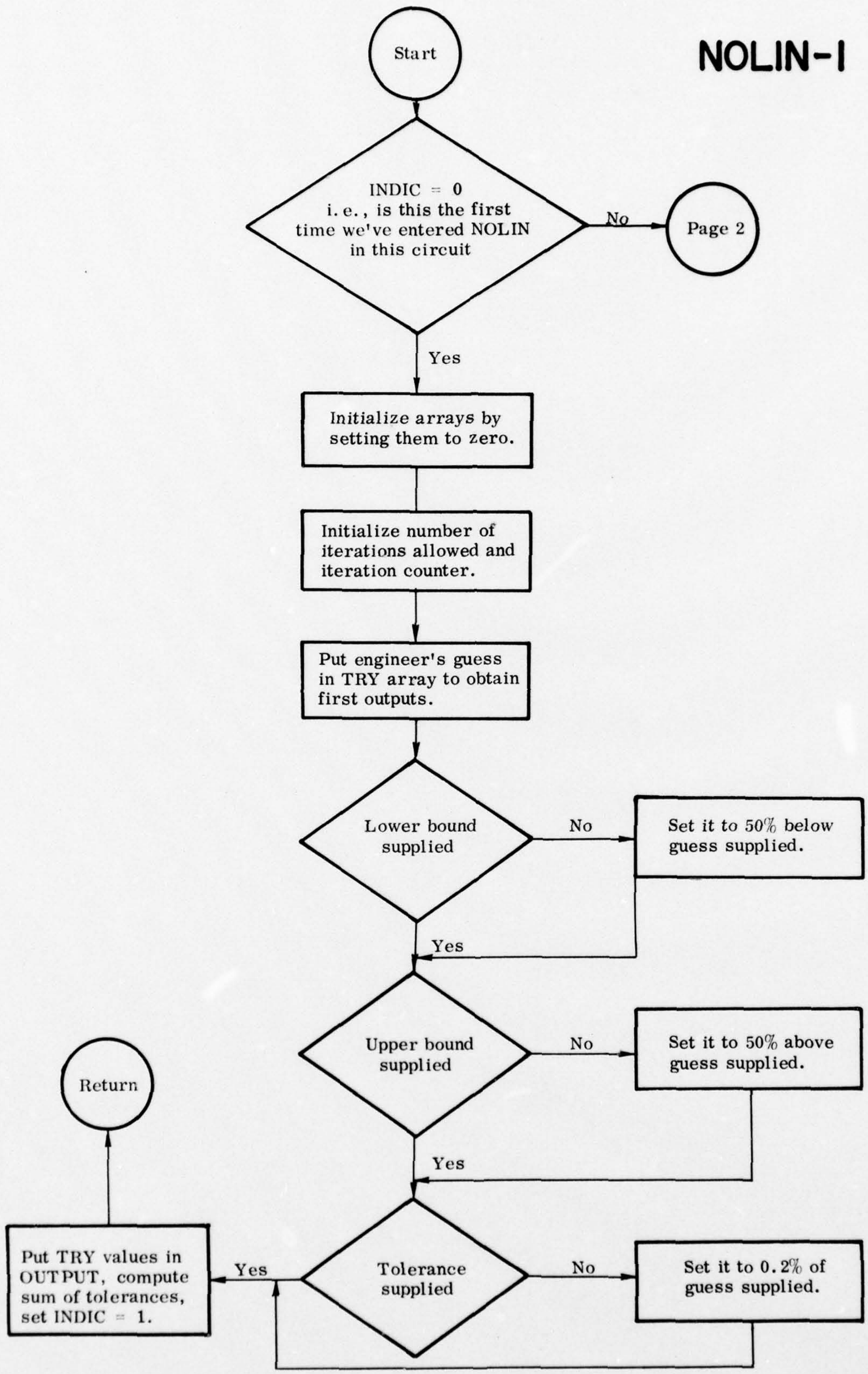

MAIN-8
Page 8-I
BRANCH on NO.
= 1
= 2
= 3
Save nominal node voltages.
Page 8-II
NO = 2.
Put low values in appropriate C. E.'s.
Page 7
Save low-value node voltages.
Set NO = 3.
Put high values in appropriate C. E.'s.
Page 7-I
Mode = 1
No
Call SENSP1.
Yes
First variation of C. E., i.e., ISN = 1
Yes
Write title, frequency, if AC.
No
Call SENSPR.
Increment ISN by 1.
Varied all JCOMP C. E.'s
No
Page 8-II
Yes
NO = 1.
Mode 1
Yes
DC circuit
No
Page 7-II
Yes
Page 2
No
BRANCH on NEXIT
= 0
= anything else
NO = 4
Page 7-I
= 5

MAIN-9



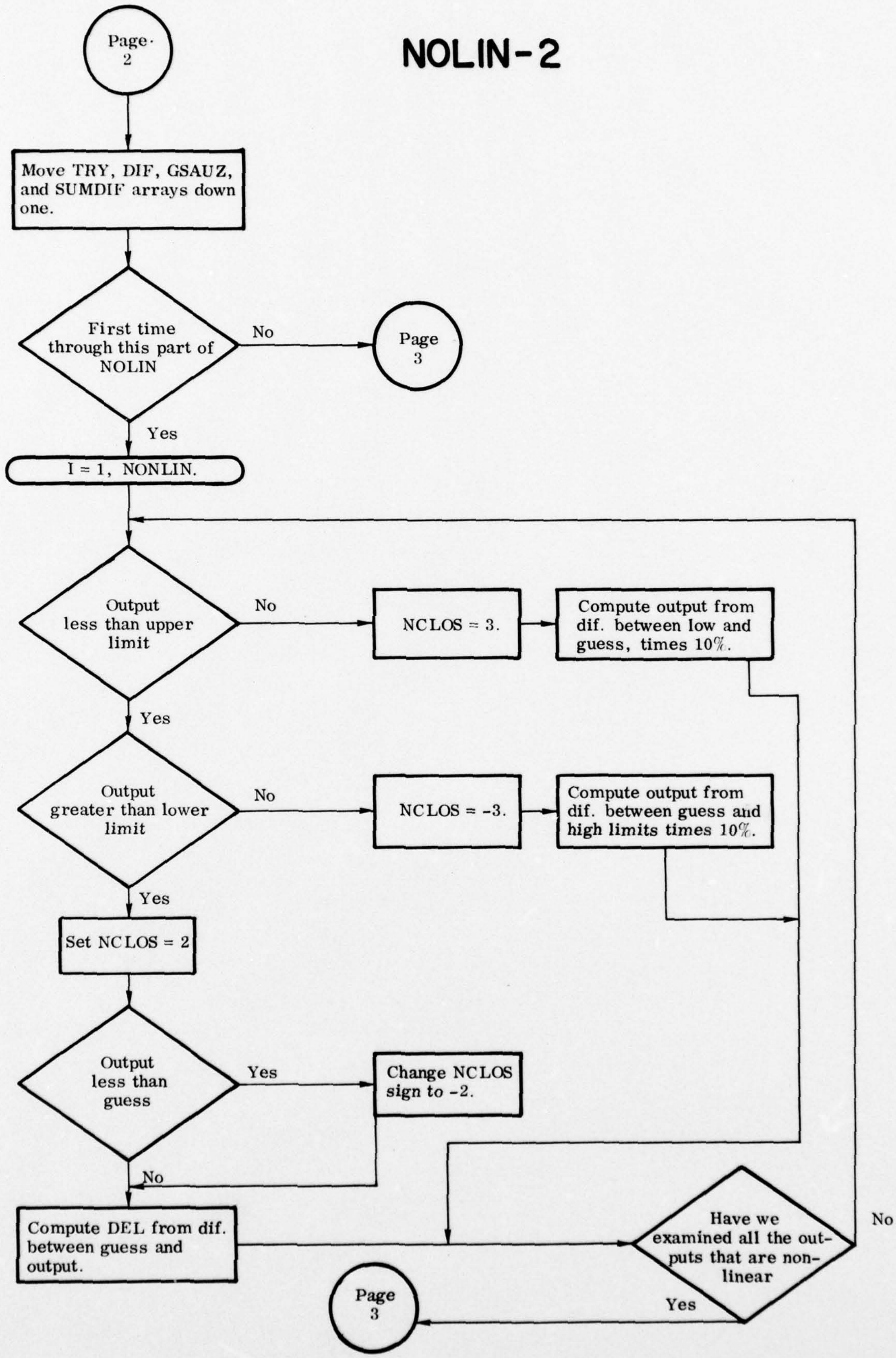
Page 9-I
Call STAT.
Increment sample counter IR by 1.
IR greater than NORV
Yes
Page 2
No
Page 4-III
Page 9-II
Call PLOTF.
Increment LF by 2.
Done for all fre- quencies
No
Page 5-III
Yes
Page 2

NOLIN-1

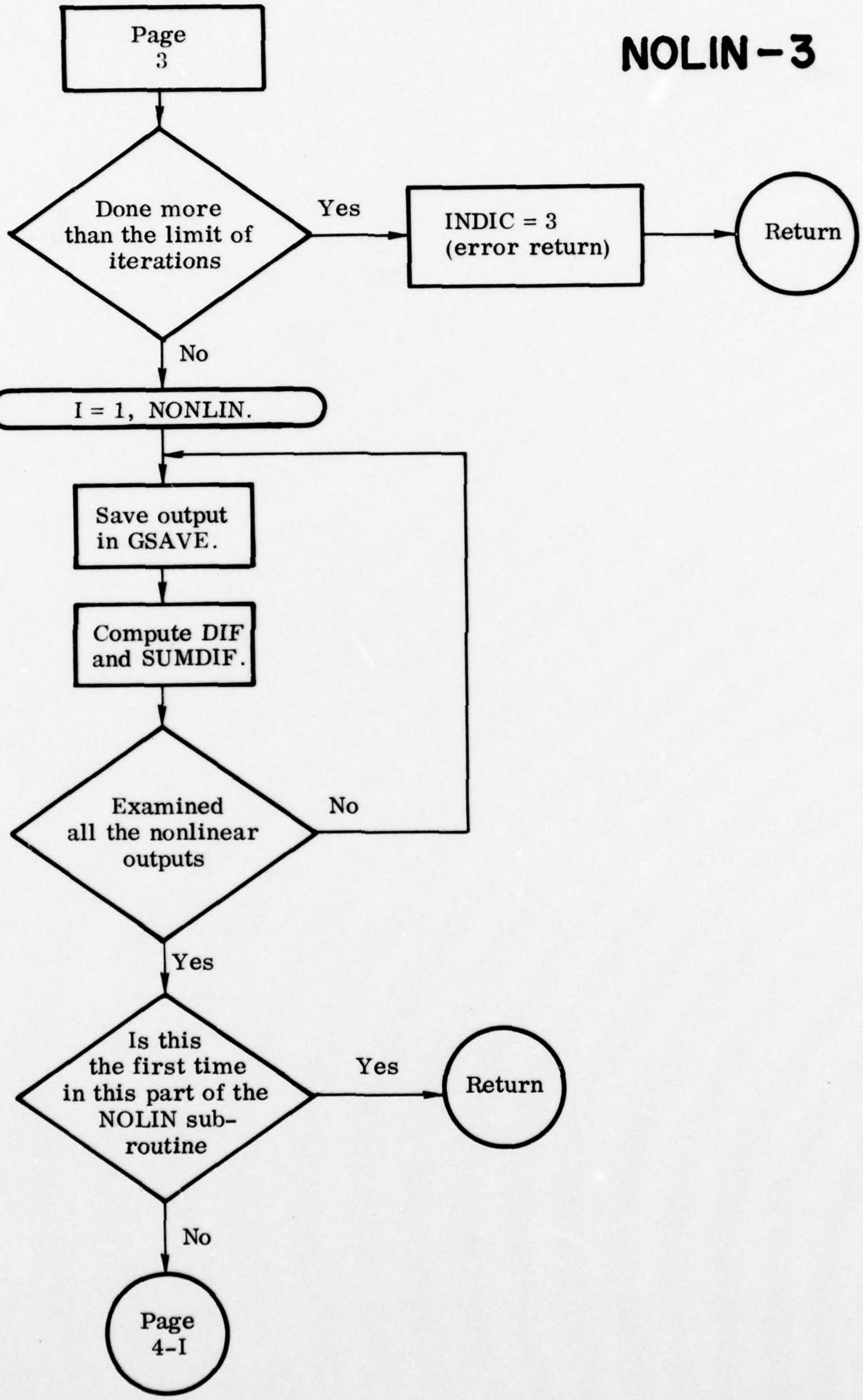


Start
INDIC = 0
i. e., is this the first
time we've entered NOLIN
in this circuit
No
Page 2
Yes
Initialize arrays by
setting them to zero.
Initialize number of
iterations allowed and
iteration counter.
Put engineer's guess
in TRY array to obtain
first outputs.
Lower bound
supplied
No
Set it to 50% below
guess supplied.
Yes
Upper bound
supplied
No
Set it to 50% above
guess supplied.
Yes
Tolerance
supplied
No
Set it to 0.2% of
guess supplied.
Yes
Put TRY values in
OUTPUT, compute
sum of tolerances,
set INDIC = 1.
Return

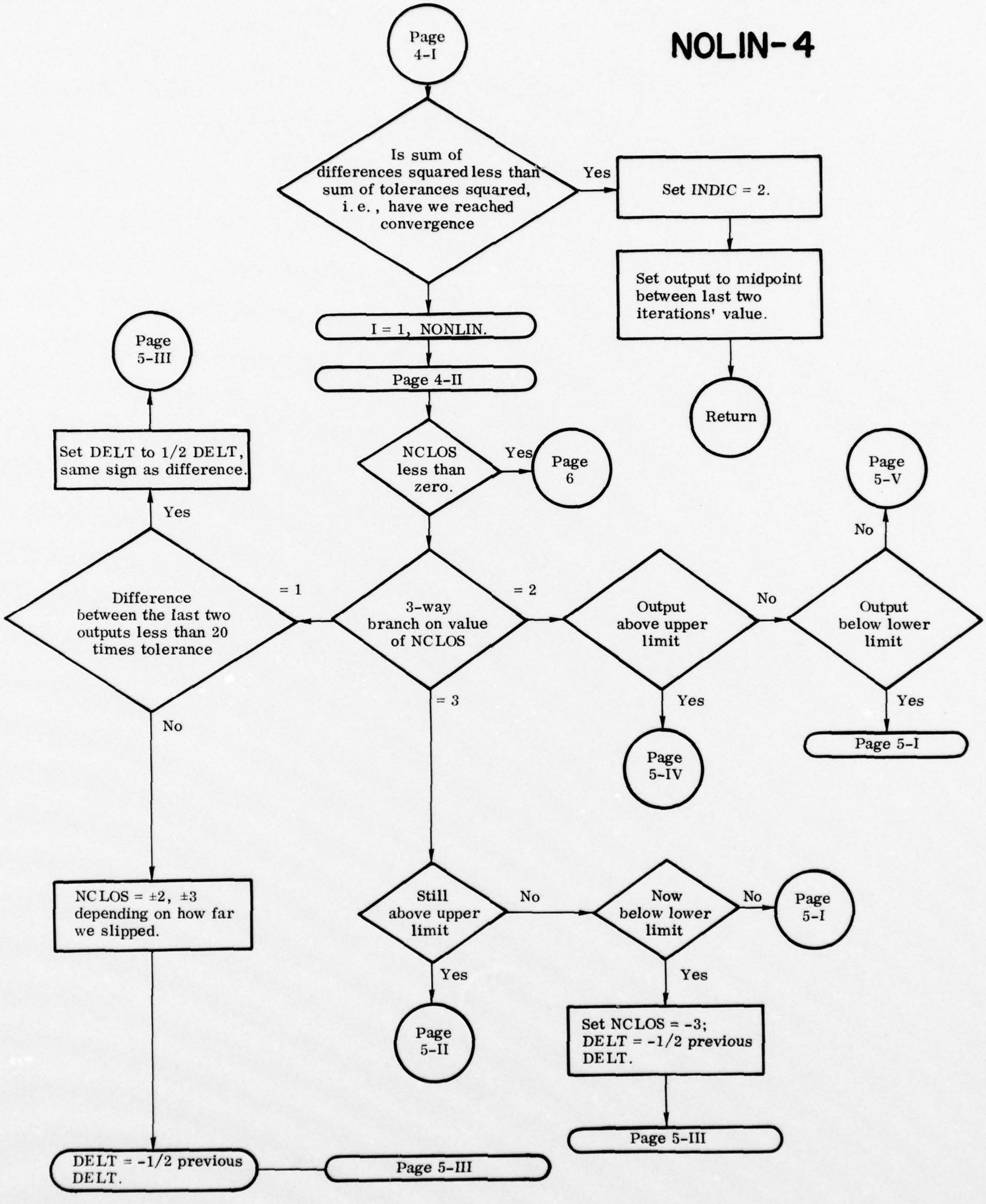
NOLIN-2


Page 2
Move TRY, DIF, GSAUZ, and SUMDIF arrays down one.
First time through this part of NOLIN
No
Page 3
Yes
I = 1, NONLIN.
Output less than upper limit
No
NCLOS = 3.
Compute output from dif. between low and guess, times 10%.
Yes
Output greater than lower limit
No
NCLOS = -3.
Compute output from dif. between guess and high limits times 10%.
Yes
Set NCLOS = 2
Output less than guess
Yes
Change NCLOS sign to -2.
No
Compute DEL from dif. between guess and output.
Have we examined all the outputs that are non-linear
No
Yes
Page 3

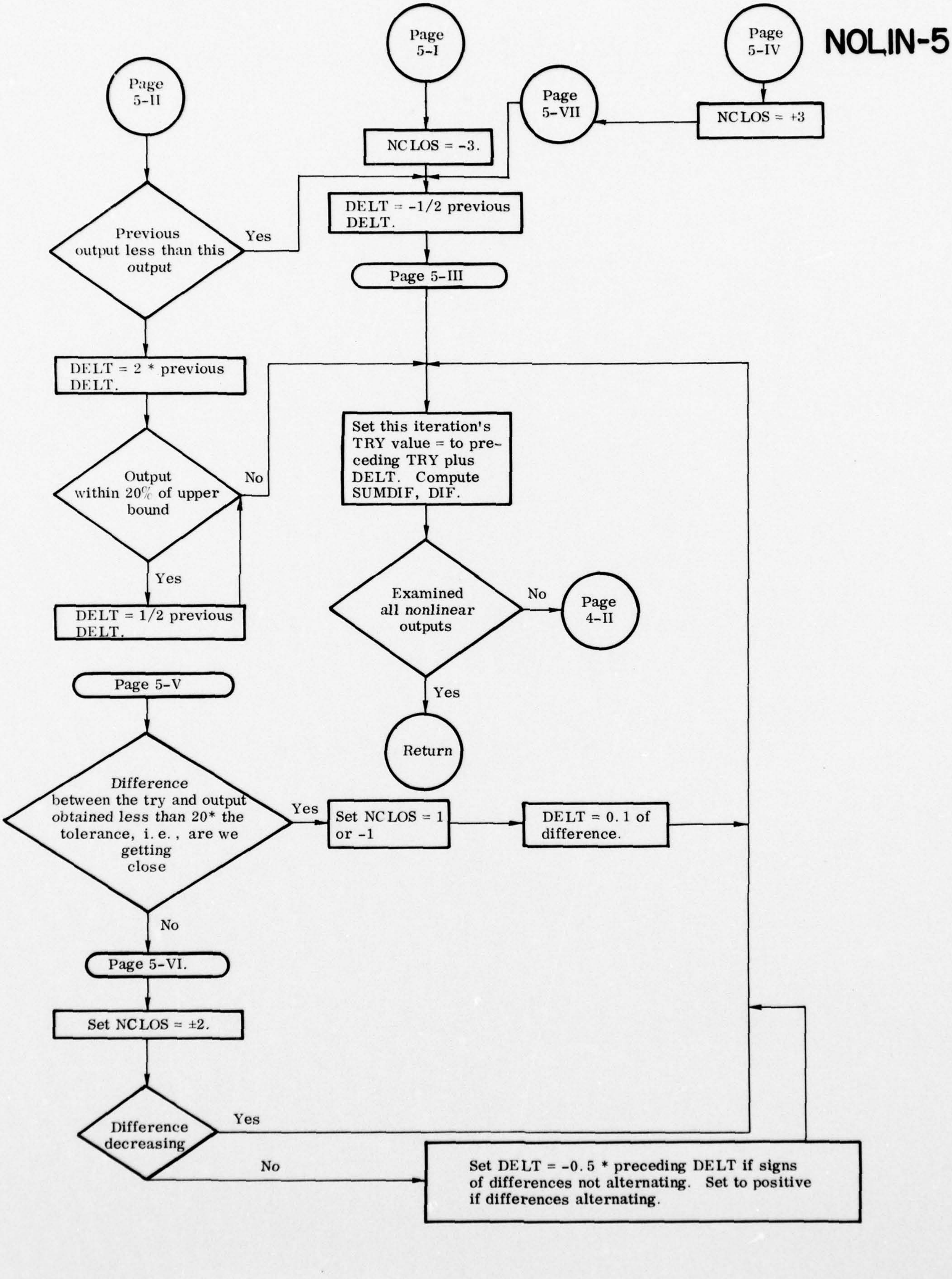
NOLIN-3


Page
3
Done more than the limit of iterations
Yes
INDIC = 3
(error return)
Return
No
I = 1, NONLIN.
Save output in GSAVE.
Compute DIF and SUMDIF.
Examined all the nonlinear outputs
No
Yes
Is this the first time in this part of the NOLIN sub-routine
Yes
Return
No
Page
4-I

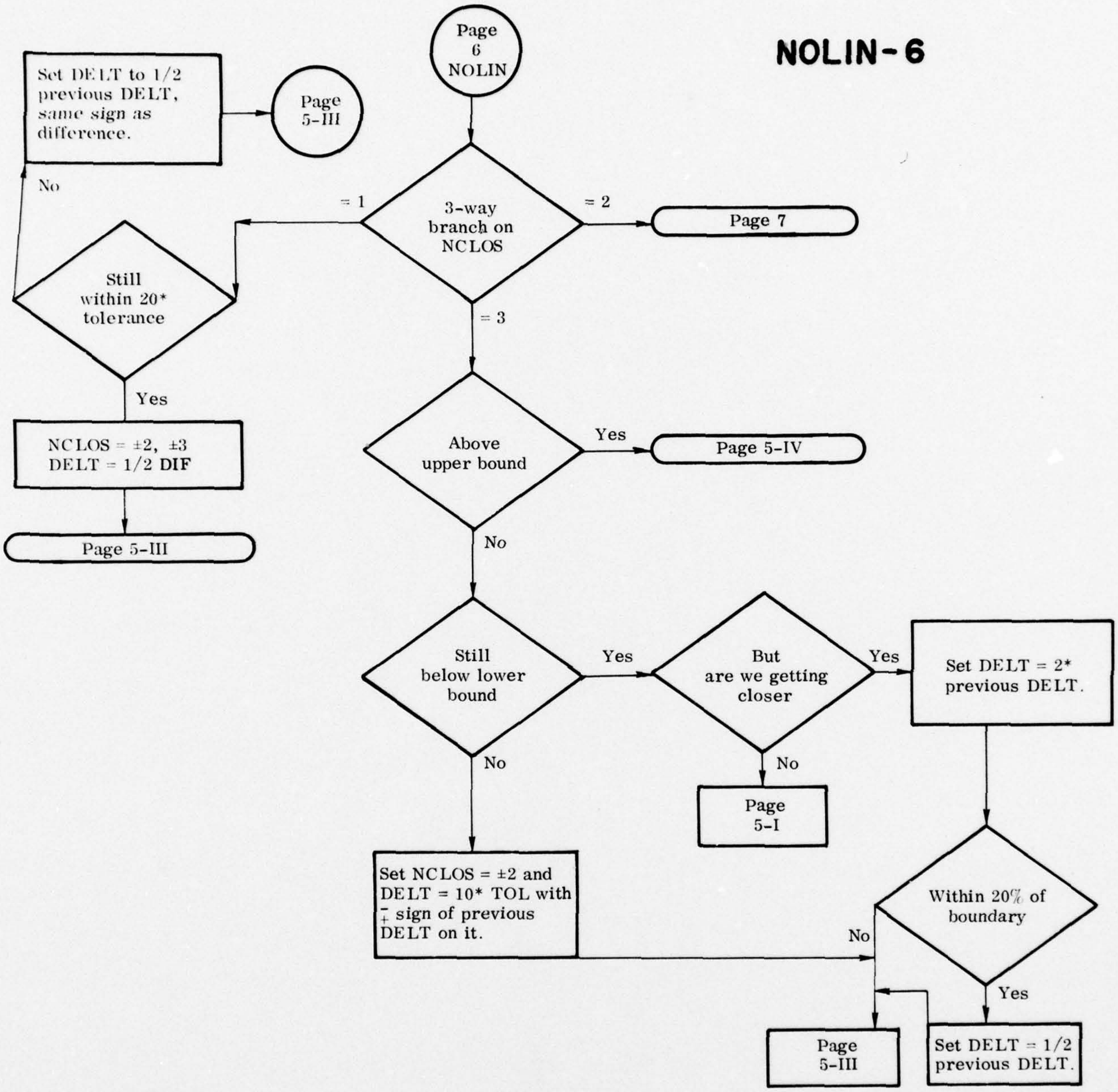
NOLIN-4



Page 4-I
Is sum of differences squared less than sum of tolerances squared, i. e., have we reached convergence
Yes
Set INDIC = 2.
Set output to midpoint between last two iterations' value.
Return
I = 1, NONLIN.
Page 4-II
NCLOS less than zero.
Yes
Page 6
3-way branch on value of NCLOS
= 1
= 2
= 3
Difference between the last two outputs less than 20 times tolerance
Yes
Set DELT to 1/2 DELT, same sign as difference.
Page 5-III
No
NCLOS = ±2, ±3 depending on how far we slipped.
DELT = -1/2 previous DELT.
Page 5-III
Output above upper limit
No
Yes
Page 5-IV
Output below lower limit
No
Page 5-V
Yes
Page 5-I
Still above upper limit
No
Yes
Page 5-II
Now below lower limit
No
Page 5-I
Yes
Set NCLOS = -3; DELT = -1/2 previous DELT.
Page 5-III

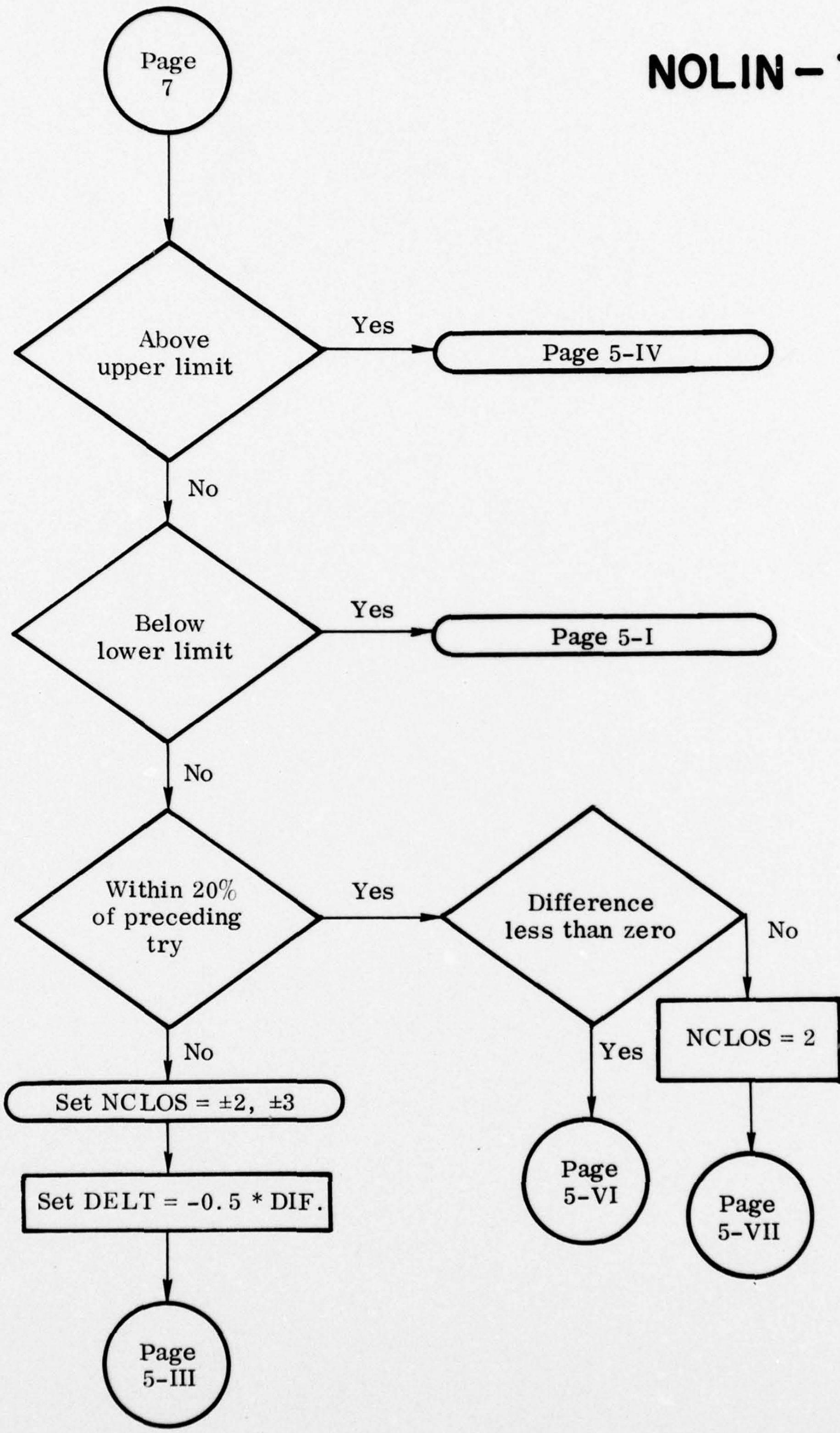

NOLIN-5
Page 5-II
Page 5-I
Page 5-IV
Page 5-VII
NCLOS = +3
NCLOS = -3.
DELT = -1/2 previous DELT.
Previous output less than this output
Yes
Page 5-III
DELT = 2 * previous DELT.
Set this iteration's TRY value = to preceding TRY plus DELT. Compute SUMDIF, DIF.
Output within 20% of upper bound
No
Yes
DELT = 1/2 previous DELT.
Examined all nonlinear outputs
No
Page 4-II
Yes
Return
Page 5-V
Difference between the try and output obtained less than 20* the tolerance, i.e., are we getting close
Yes
Set NCLOS = 1 or -1
DELT = 0.1 of difference.
No
Page 5-VI.
Set NCLOS = ±2.
Difference decreasing
Yes
No
Set DELT = -0.5 * preceding DELT if signs of differences not alternating. Set to positive if differences alternating.

NOLIN-6

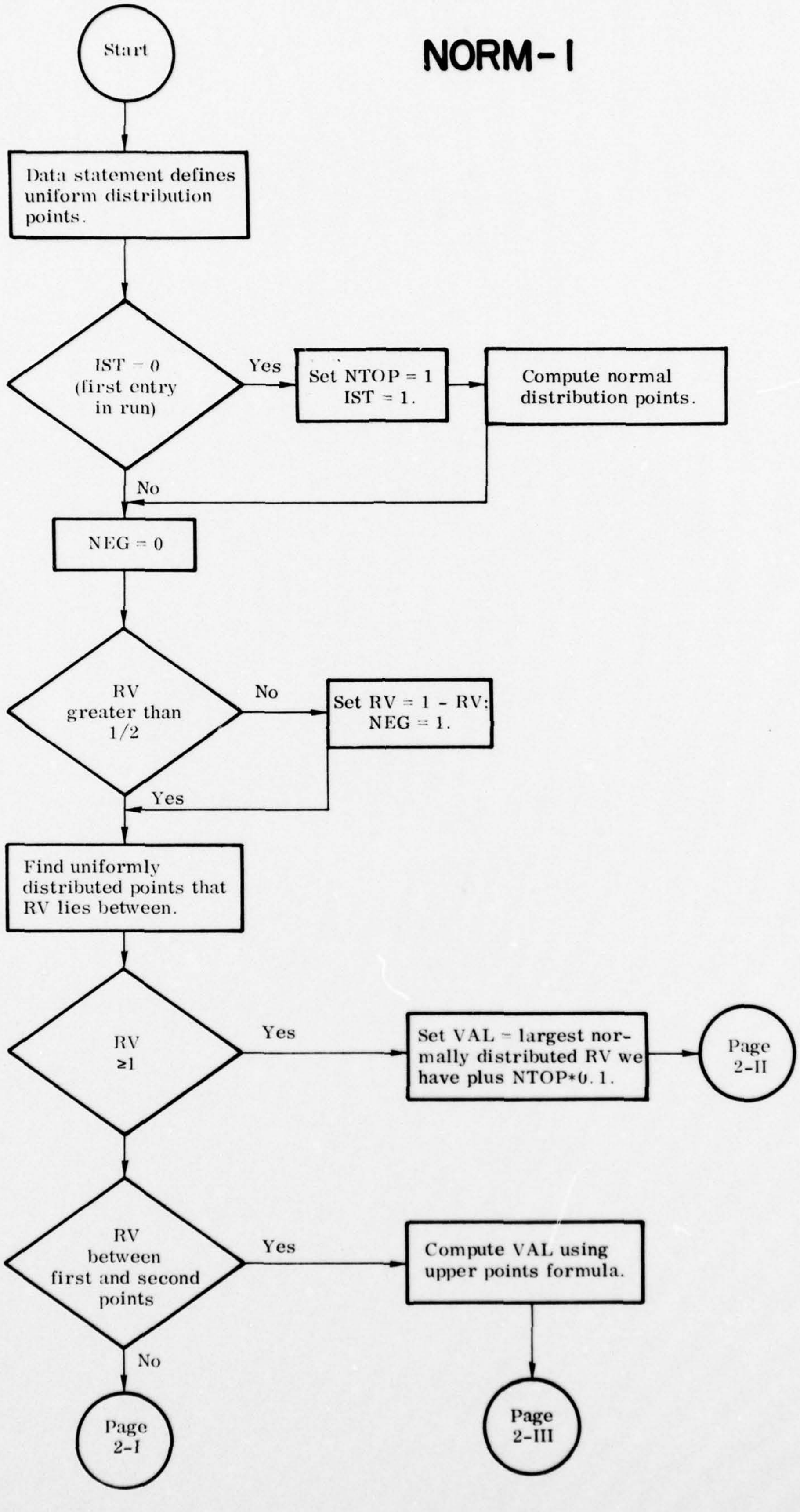


Page
6
NOLIN
3-way
branch on
NCLOS
= 1
= 2
= 3
Page 7
Set DELT to 1/2
previous DELT,
same sign as
difference.
Page
5-III
No
Still
within 20*
tolerance
Yes
NCLOS = ±2, ±3
DELT = 1/2 DIF
Page 5-III
Above
upper bound
Yes
Page 5-IV
No
Still
below lower
bound
Yes
But
are we getting
closer
Yes
Set DELT = 2*
previous DELT.
No
Page
5-I
No
Set NCLOS = ±2 and
DELT = 10* TOL with
∓ sign of previous
DELT on it.
Within 20% of
boundary
No
Yes
Page
5-III
Set DELT = 1/2
previous DELT.

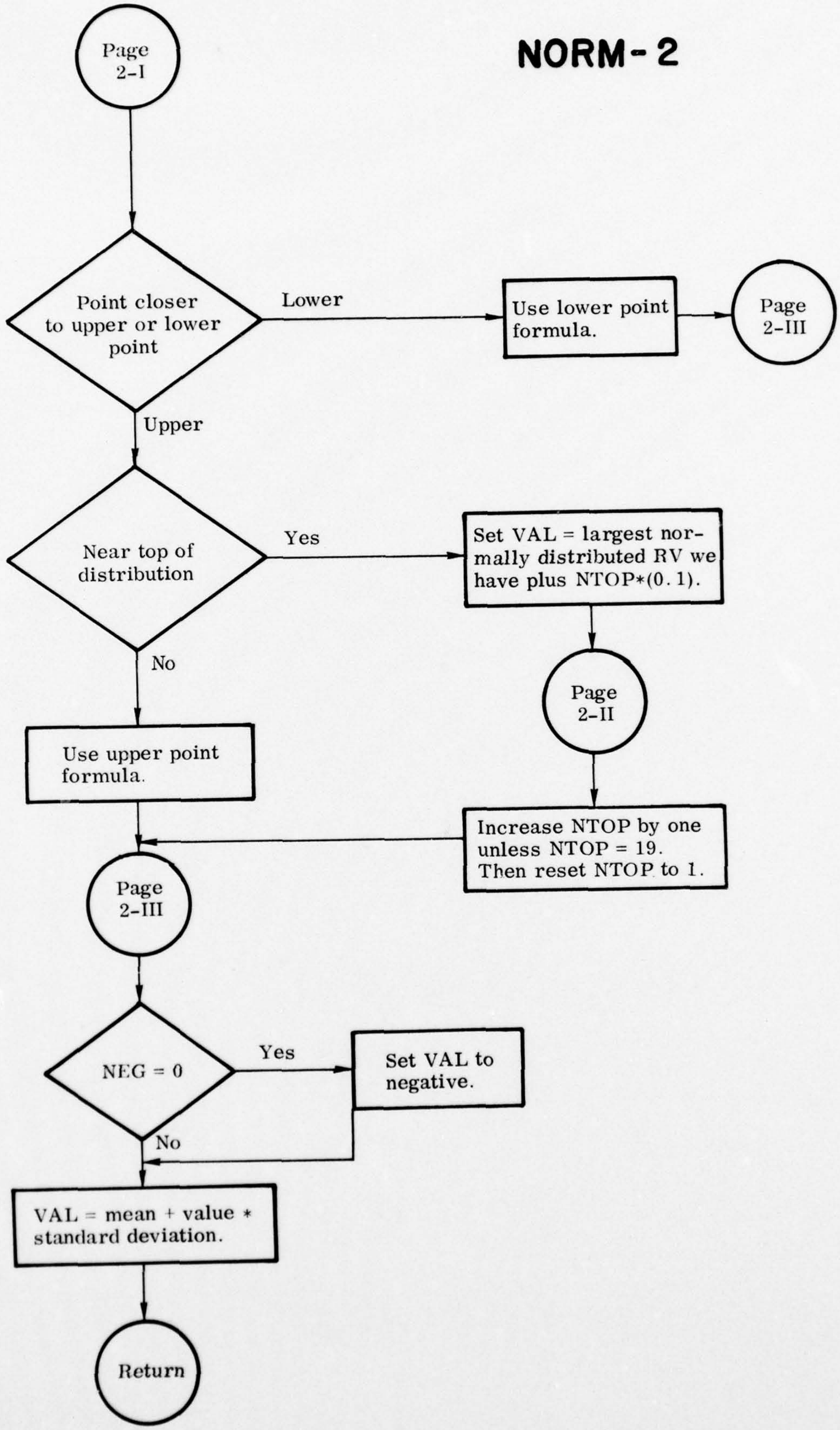
NOLIN - 7


Page
7
Above
upper limit
Yes
Page 5-IV
No
Below
lower limit
Yes
Page 5-I
No
Within 20%
of preceding
try
Yes
Difference
less than zero
No
No
Set NCLOS = ±2, ±3
Set DELT = -0.5 * DIF.
Page
5-III
Yes
NCLOS = 2
Page
5-VI
Page
5-VII

NORM-I

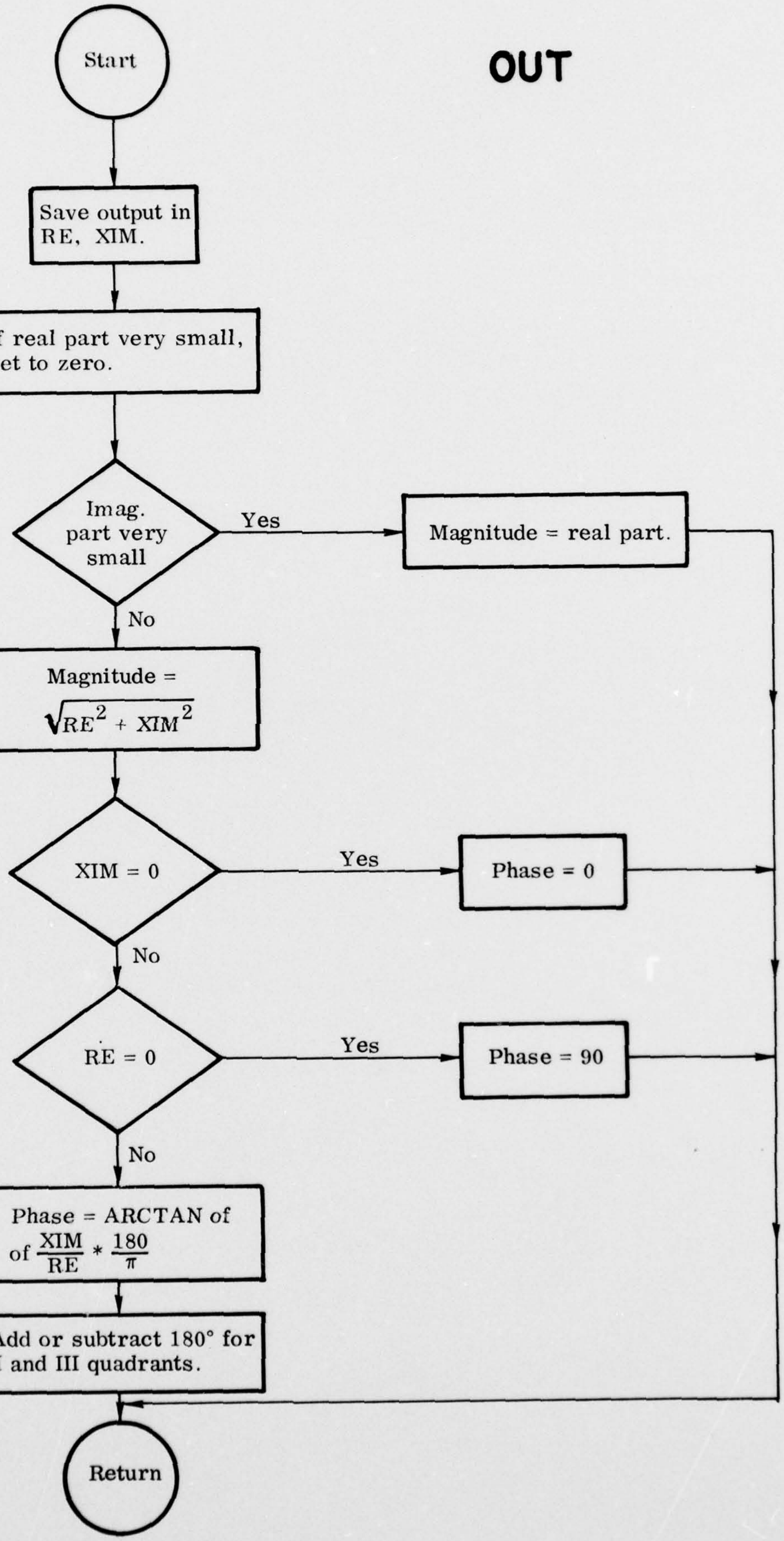

Start
Data statement defines uniform distribution points.
IST = 0 (first entry in run)
Yes
Set NTOP = 1 IST = 1.
Compute normal distribution points.
No
NEG = 0
RV greater than 1/2
No
Set RV = 1 - RV: NEG = 1.
Yes
Find uniformly distributed points that RV lies between.
RV ≥1
Yes
Set VAL = largest normally distributed RV we have plus NTOP*0.1.
Page 2-II
RV between first and second points
Yes
Compute VAL using upper points formula.
No
Page 2-I
Page 2-III

NORM-2

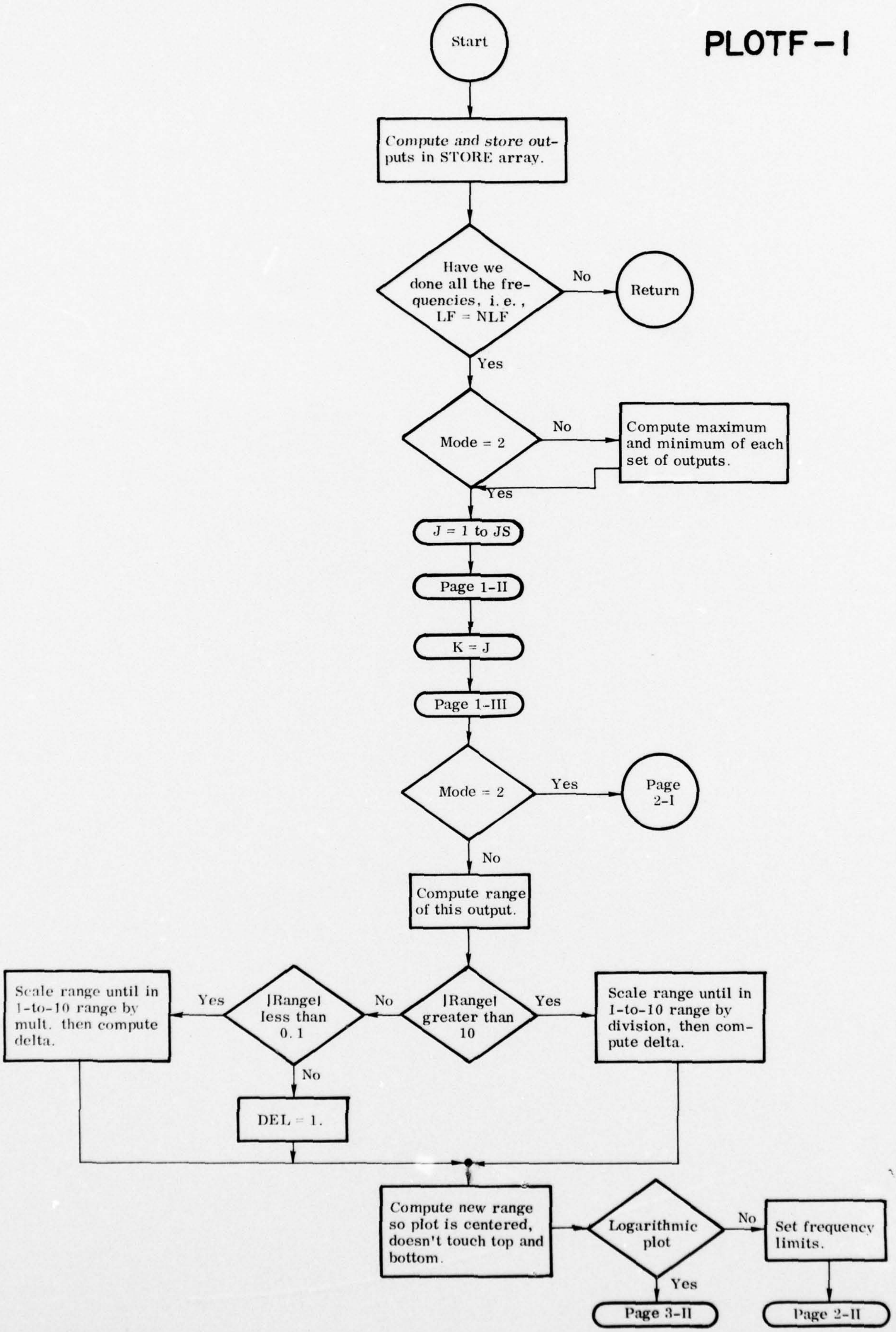


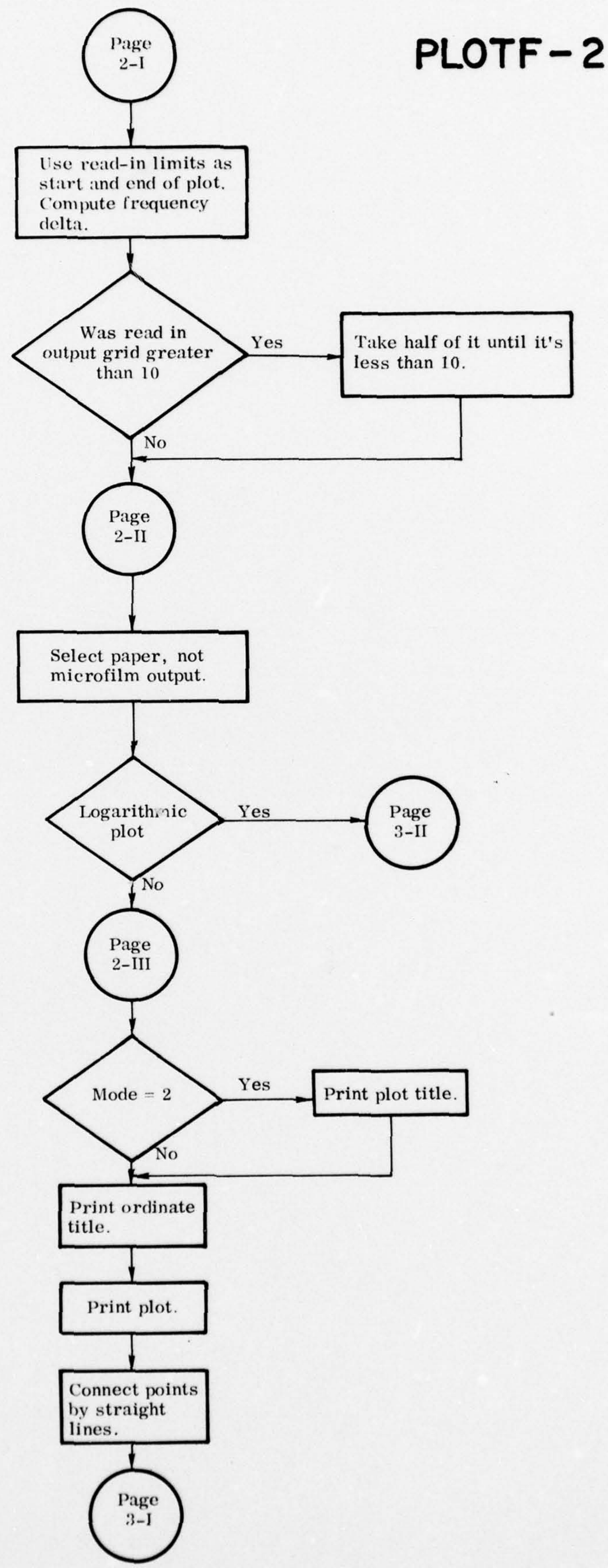
Page 2-I
Point closer to upper or lower point
Lower
Use lower point formula.
Page 2-III
Upper
Near top of distribution
Yes
Set VAL = largest normally distributed RV we have plus NTOP*(0.1).
Page 2-II
No
Use upper point formula.
Increase NTOP by one unless NTOP = 19. Then reset NTOP to 1.
Page 2-III
NEG = 0
Yes
Set VAL to negative.
No
VAL = mean + value * standard deviation.
Return

OUT

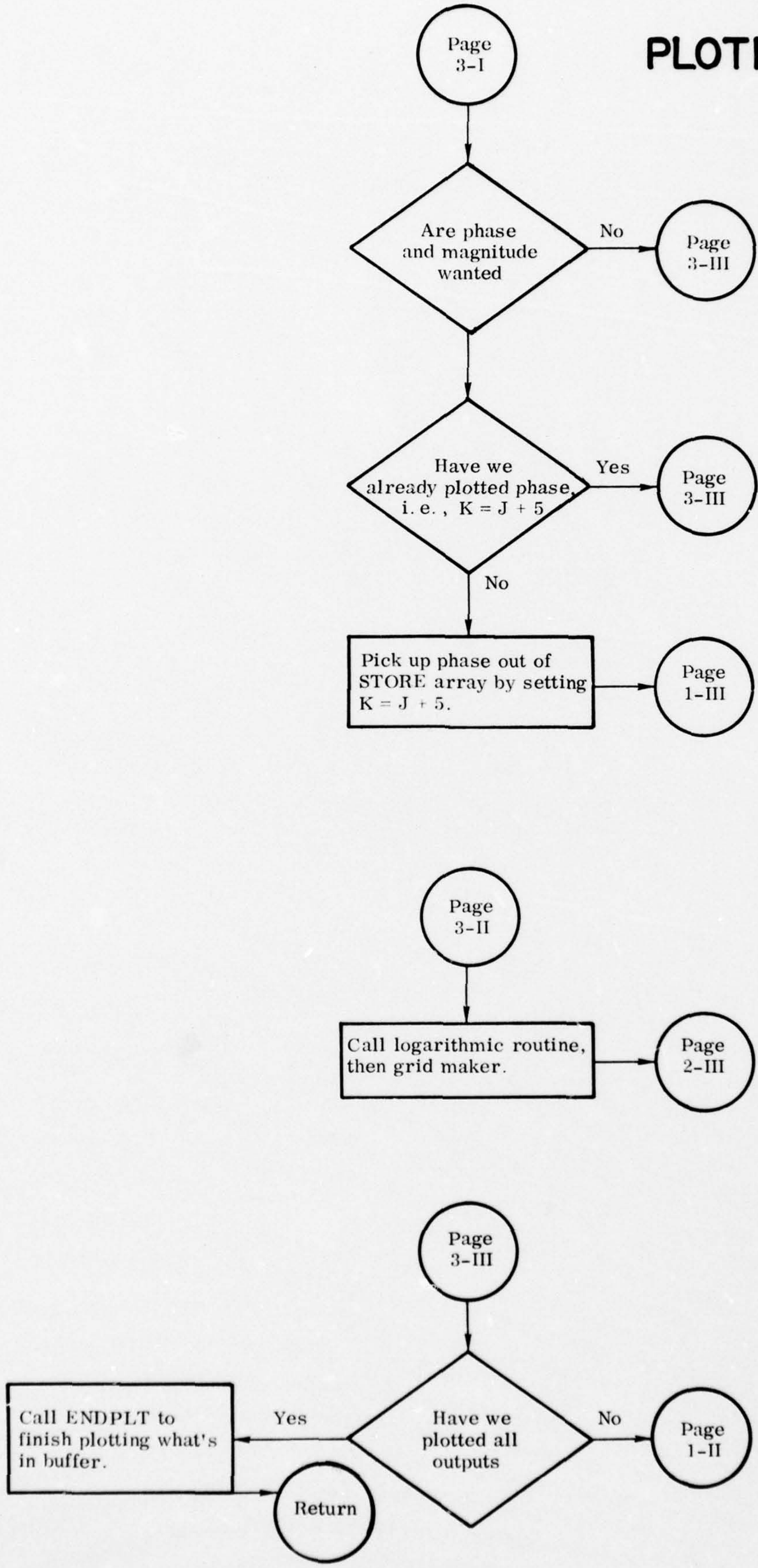

Start
Save output in RE, XIM.
If real part very small, set to zero.
Imag. part very small
Yes
Magnitude = real part.
No
Magnitude = $\sqrt{RE^2 + XIM^2}$
XIM = 0
Yes
Phase = 0
No
RE = 0
Yes
Phase = 90
No
Phase = ARCTAN of of $\frac{XIM}{RE} * \frac{180}{\pi}$
Add or subtract 180° for II and III quadrants.
Return

PLOTF-1


Start
Compute and store outputs in STORE array.
Have we done all the frequencies, i. e., LF = NLF
No
Return
Yes
Mode = 2
No
Compute maximum and minimum of each set of outputs.
Yes
J = 1 to JS
Page 1-II
K = J
Page 1-III
Mode = 2
Yes
Page 2-I
No
Compute range of this output.
|Range| greater than 10
Yes
Scale range until in 1-to-10 range by division, then compute delta.
No
|Range| less than 0.1
Yes
Scale range until in 1-to-10 range by mult. then compute delta.
No
DEL = 1.
Compute new range so plot is centered, doesn't touch top and bottom.
Logarithmic plot
No
Set frequency limits.
Yes
Page 3-II
Page 2-II

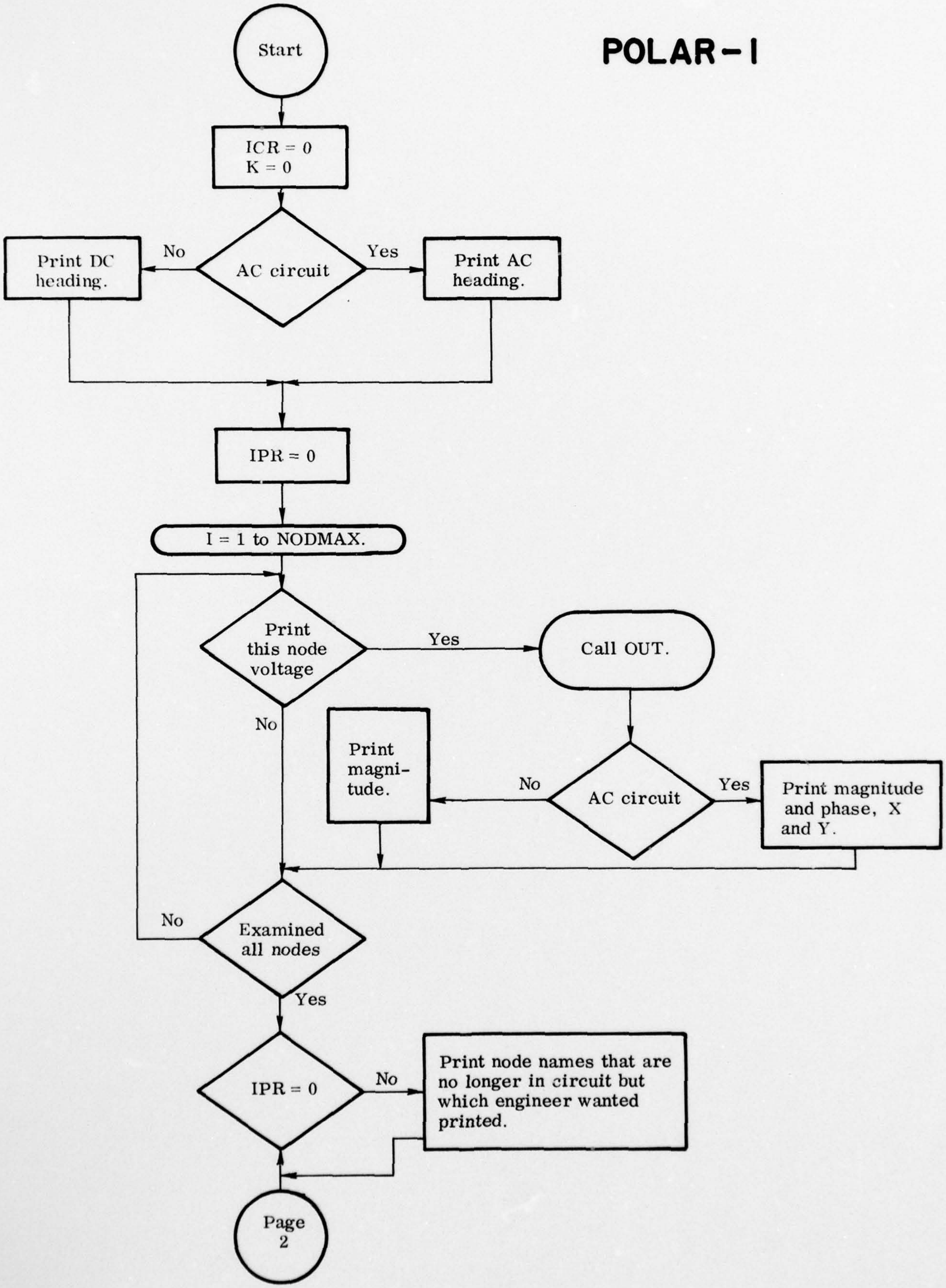

PLOTF-2
Page 2-I
Use read-in limits as start and end of plot. Compute frequency delta.
Was read in output grid greater than 10
Yes
Take half of it until it's less than 10.
No
Page 2-II
Select paper, not microfilm output.
Logarithmic plot
Yes
Page 3-II
No
Page 2-III
Mode = 2
Yes
Print plot title.
No
Print ordinate title.
Print plot.
Connect points by straight lines.
Page 3-I

PLOTF - 3



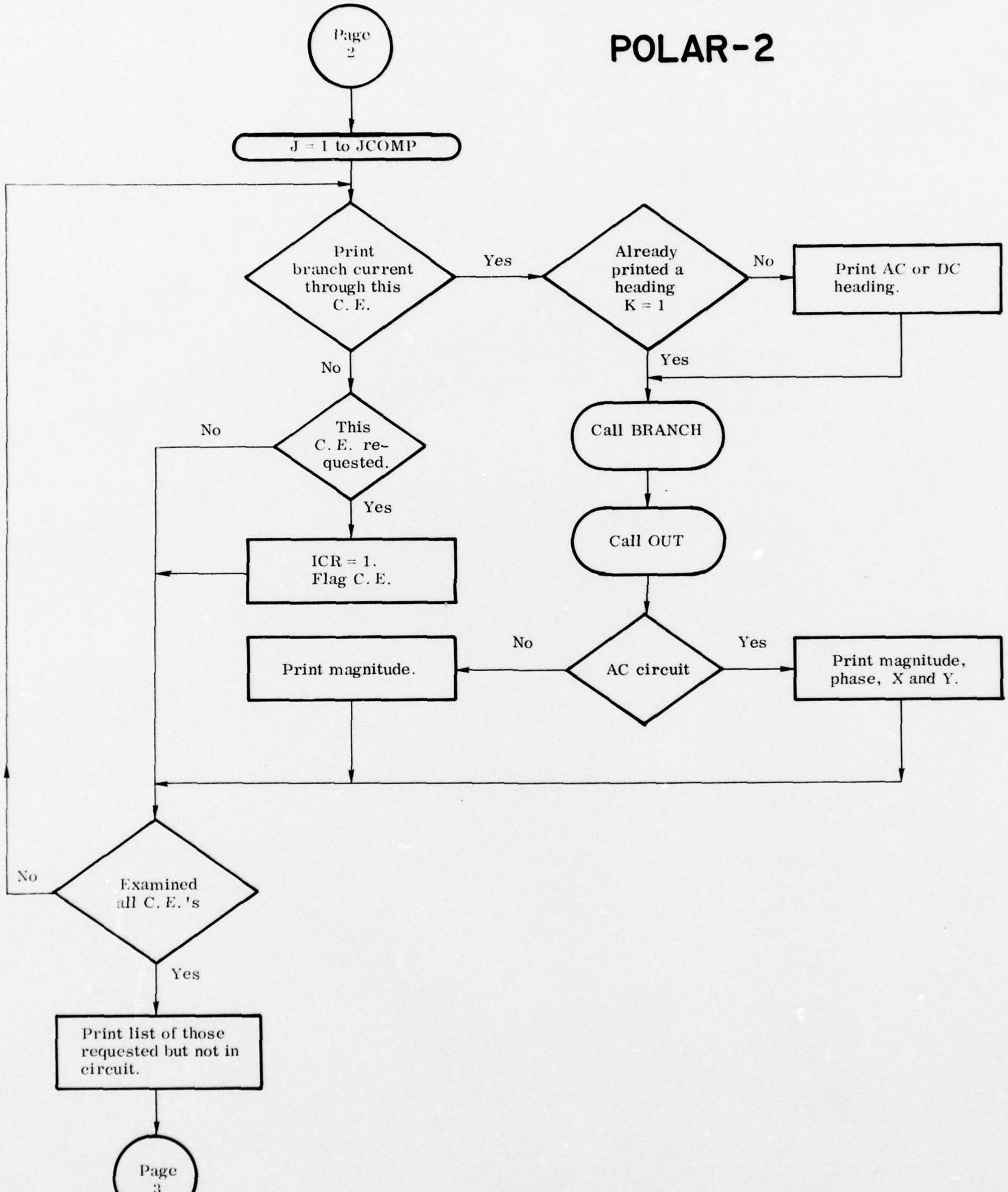
Page 3-I
Are phase and magnitude wanted
No
Page 3-III
Have we already plotted phase, i.e., K = J + 5
Yes
Page 3-III
No
Pick up phase out of STORE array by setting K = J + 5.
Page 1-III
Page 3-II
Call logarithmic routine, then grid maker.
Page 2-III
Page 3-III
Have we plotted all outputs
Yes
Call ENDPLT to finish plotting what's in buffer.
Return
No
Page 1-II

POLAR-1

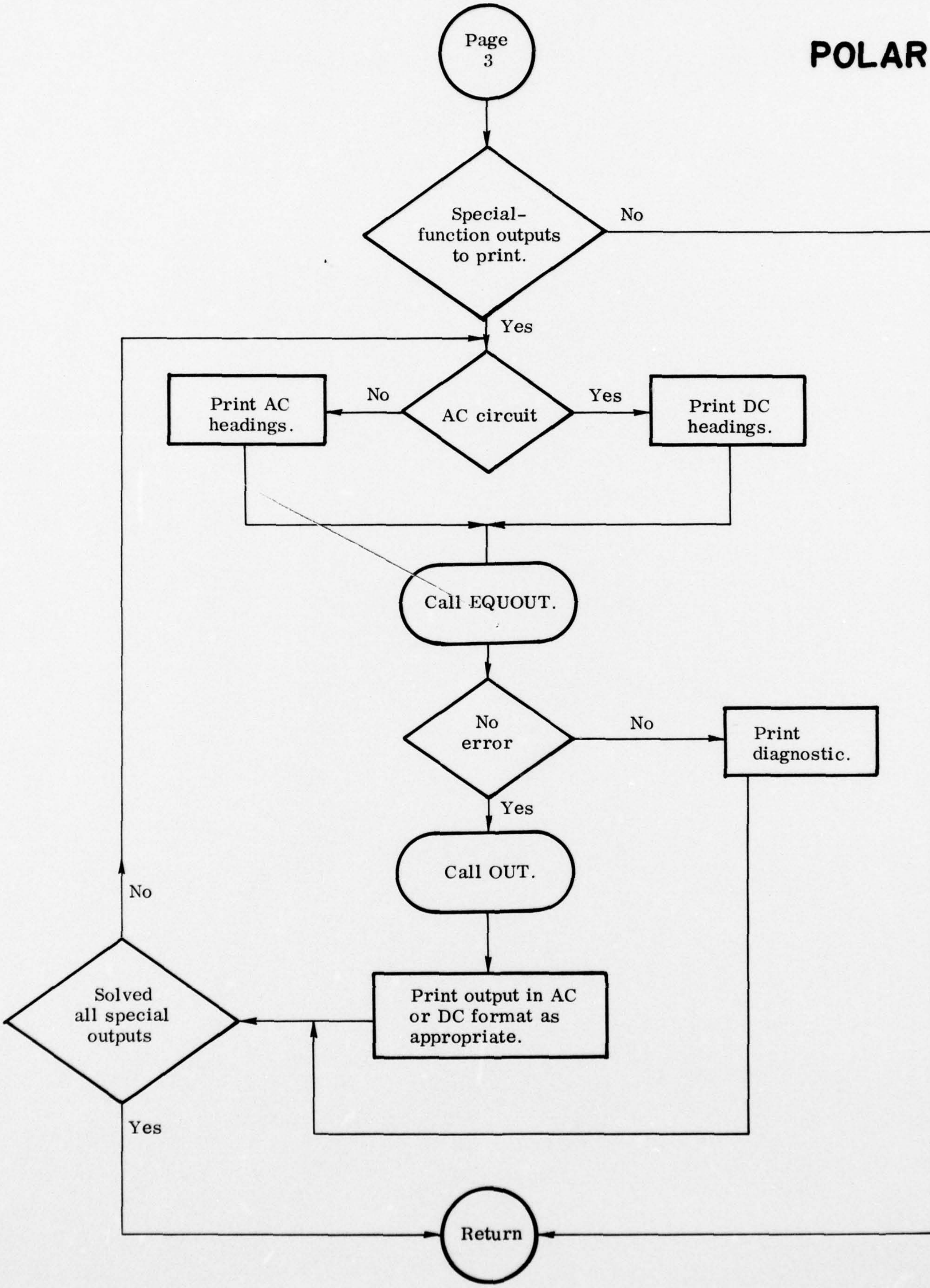


Start
ICR = 0
K = 0
AC circuit
No
Yes
Print DC heading.
Print AC heading.
IPR = 0
I = 1 to NODMAX.
Print this node voltage
Yes
Call OUT.
No
AC circuit
No
Yes
Print magnitude.
Print magnitude and phase, X and Y.
Examined all nodes
No
Yes
IPR = 0
No
Print node names that are no longer in circuit but which engineer wanted printed.
Page 2

POLAR-2

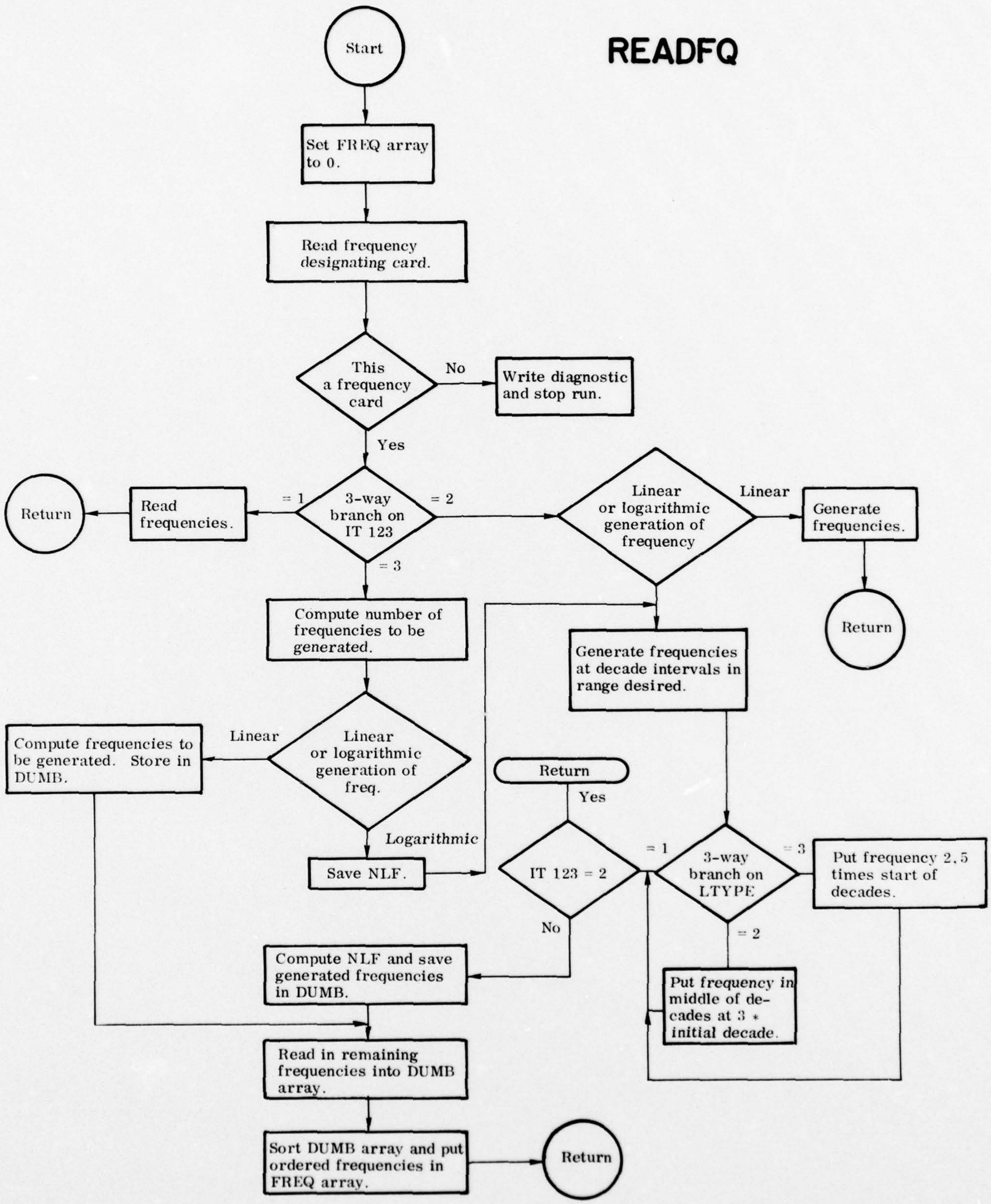

Page 2
J = 1 to JCOMP
Print branch current through this C. E.
Yes
Already printed a heading K = 1
No
Print AC or DC heading.
Yes
No
This C. E. re-quested.
No
Yes
ICR = 1. Flag C. E.
Call BRANCH
Call OUT
AC circuit
No
Yes
Print magnitude.
Print magnitude, phase, X and Y.
Examined all C. E.'s
No
Yes
Print list of those requested but not in circuit.
Page 3

POLAR - 3

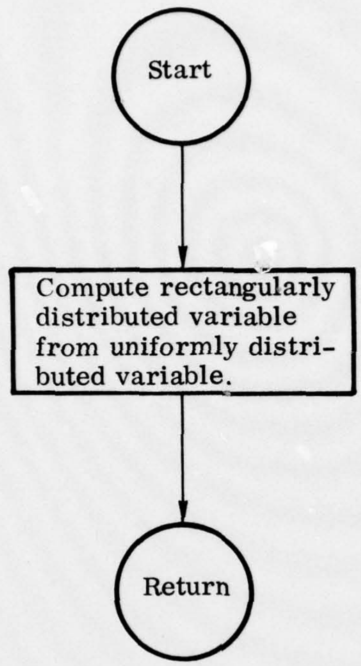


Page
3
Special-
function outputs
to print.
No
Yes
No
Print AC
headings.
AC circuit
Yes
Print DC
headings.
Call EQUOUT.
No
error
No
Print
diagnostic.
Yes
Call OUT.
No
Solved
all special
outputs
Print output in AC
or DC format as
appropriate.
Yes
Return

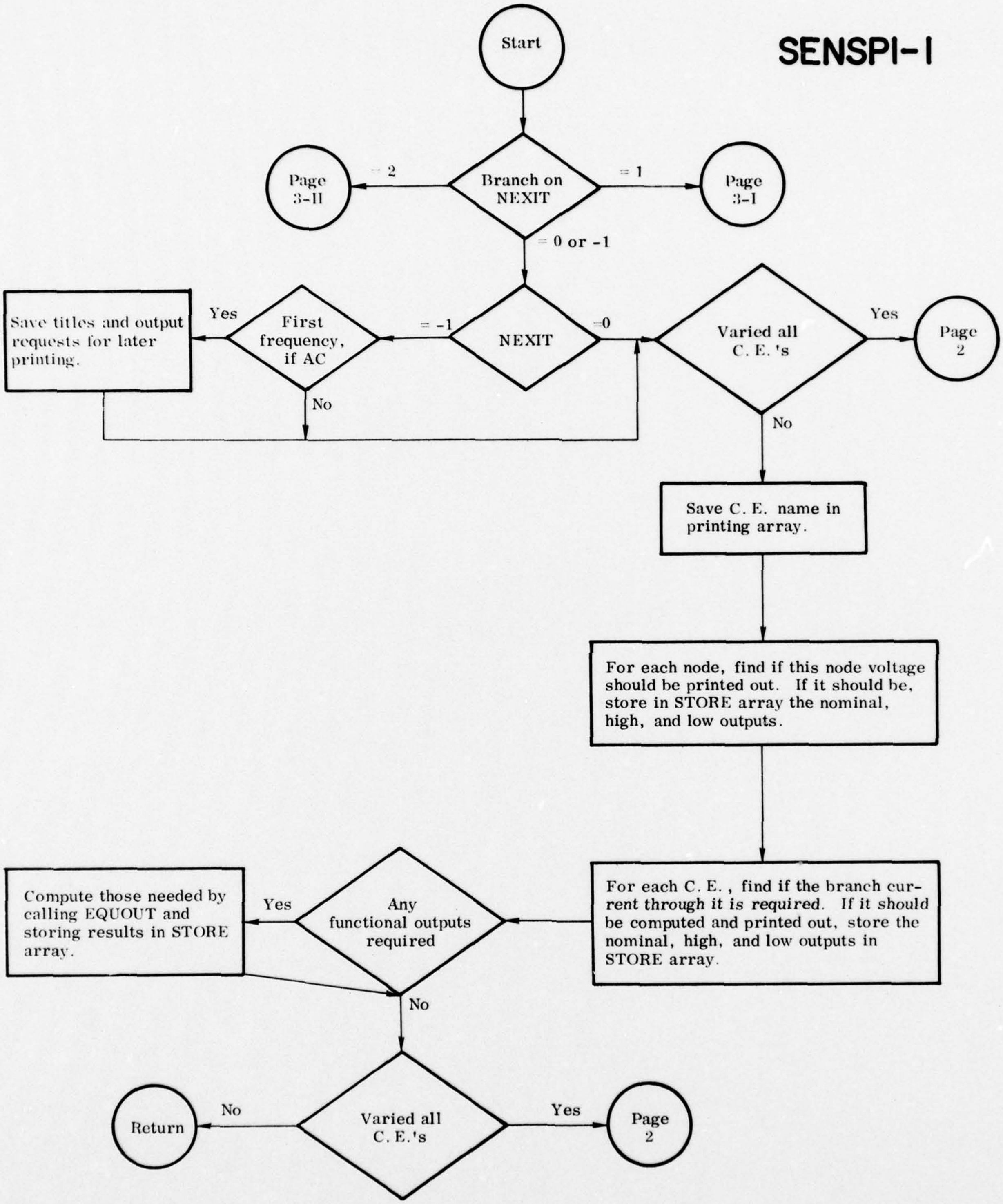
READFQ


Start
Set FREQ array to 0.
Read frequency designating card.
This a frequency card
No
Write diagnostic and stop run.
Yes
3-way branch on IT 123
= 1
Read frequencies.
Return
= 2
Linear or logarithmic generation of frequency
Linear
Generate frequencies.
Return
= 3
Compute number of frequencies to be generated.
Generate frequencies at decade intervals in range desired.
Linear
Compute frequencies to be generated. Store in DUMB.
Linear or logarithmic generation of freq.
Logarithmic
Save NLF.
Return
Yes
IT 123 = 2
No
= 1
3-way branch on LTYPE
= 3
Put frequency 2.5 times start of decades.
= 2
Put frequency in middle of decades at 3 * initial decade.
Compute NLF and save generated frequencies in DUMB.
Read in remaining frequencies into DUMB array.
Sort DUMB array and put ordered frequencies in FREQ array.
Return

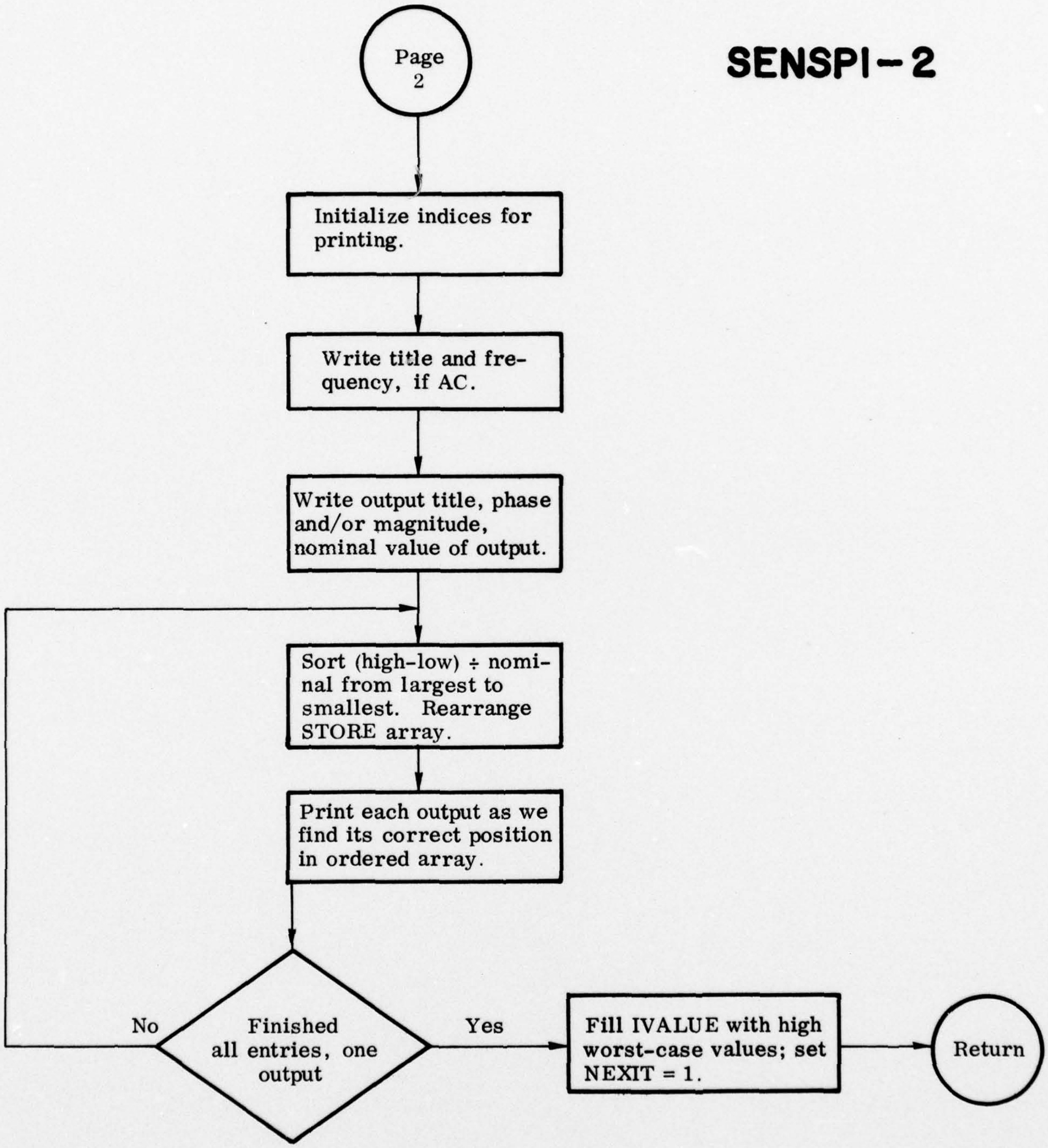
RECTAN


Start
Compute rectangularly distributed variable from uniformly distributed variable.
Return

SENSPI-1

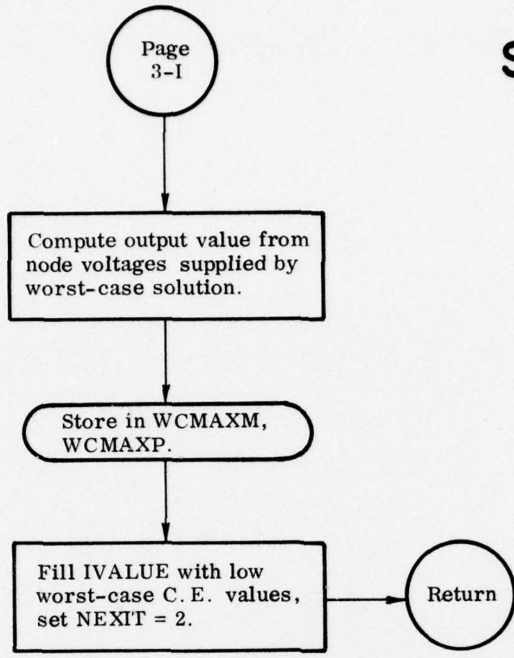

Start
Branch on NEXIT
= 2
Page 3-II
= 1
Page 3-I
= 0 or -1
NEXIT
= -1
First frequency, if AC
Yes
Save titles and output requests for later printing.
No
=0
Varied all C. E.'s
Yes
Page 2
No
Save C. E. name in printing array.
For each node, find if this node voltage should be printed out. If it should be, store in STORE array the nominal, high, and low outputs.
For each C. E., find if the branch current through it is required. If it should be computed and printed out, store the nominal, high, and low outputs in STORE array.
Any functional outputs required
Yes
Compute those needed by calling EQUOUT and storing results in STORE array.
No
Varied all C. E.'s
No
Return
Yes
Page 2

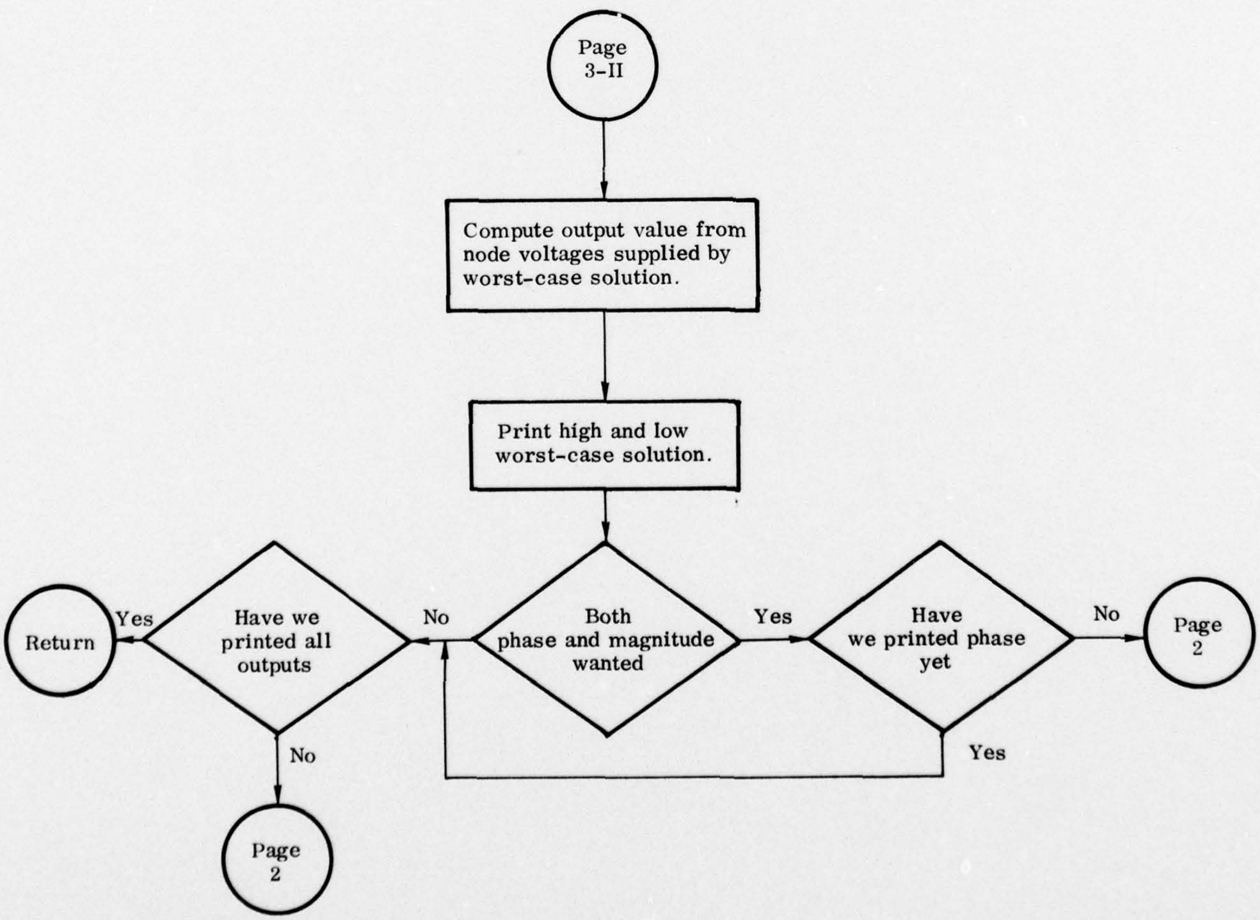
SENSPI-2



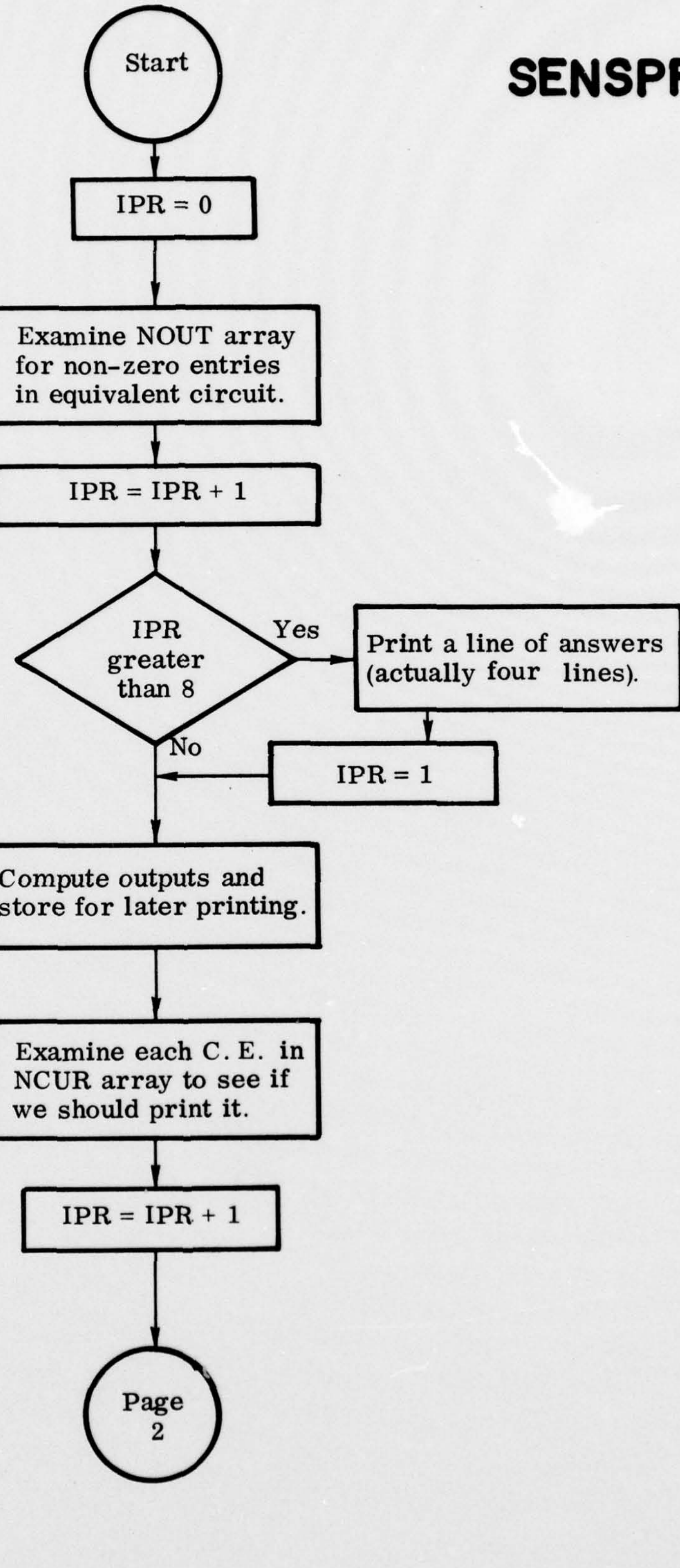
Page 2
Initialize indices for printing.
Write title and frequency, if AC.
Write output title, phase and/or magnitude, nominal value of output.
Sort (high-low) ÷ nominal from largest to smallest. Rearrange STORE array.
Print each output as we find its correct position in ordered array.
Finished all entries, one output
No
Yes
Fill IVALUE with high worst-case values; set NEXIT = 1.
Return

SENSPI-3

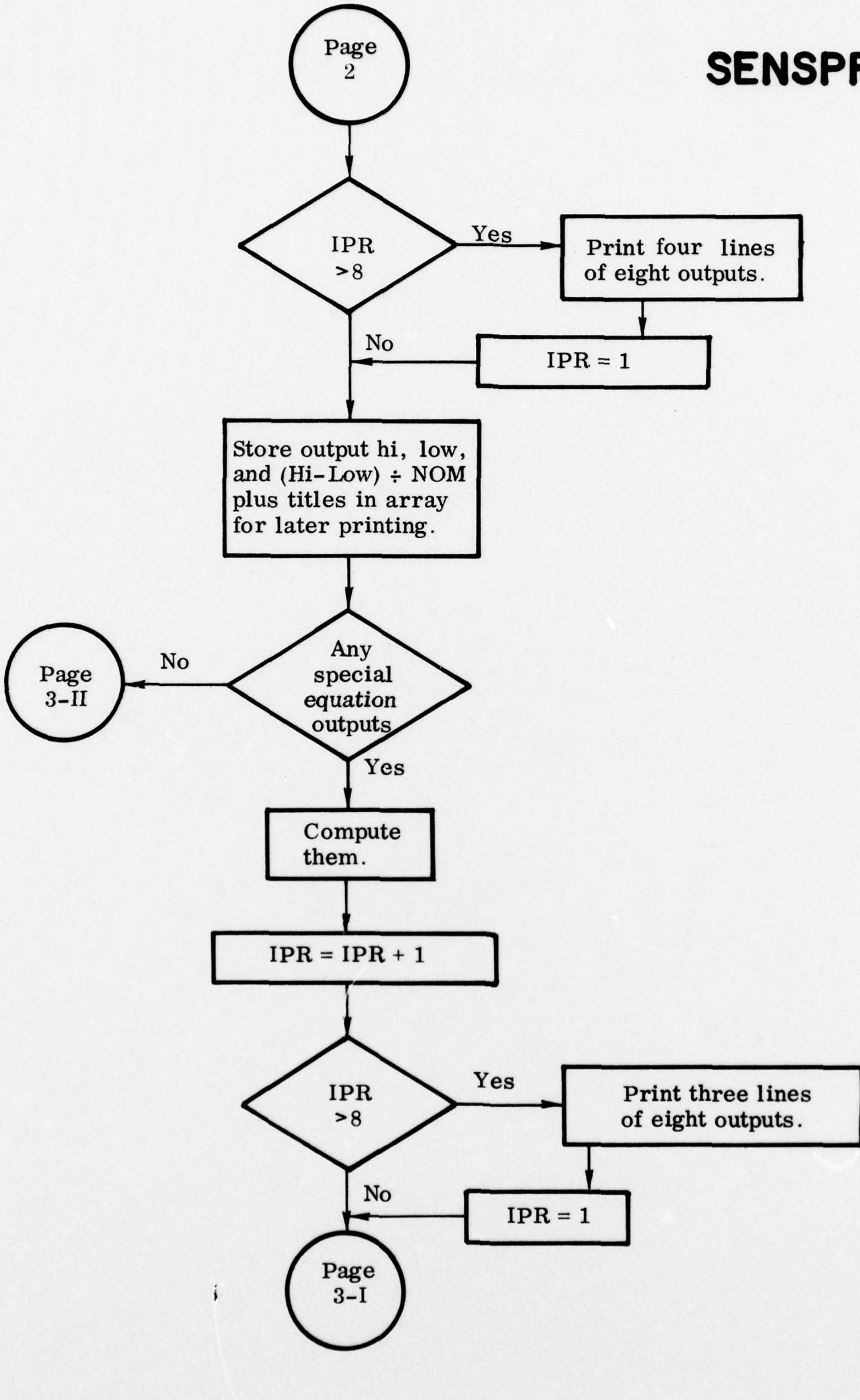

Page 3-I
Compute output value from node voltages supplied by worst-case solution.
Store in WCMAXM, WCMAXP.
Fill IVALUE with low worst-case C.E. values, set NEXIT = 2.
Return


Page 3-II
Compute output value from node voltages supplied by worst-case solution.
Print high and low worst-case solution.
Both phase and magnitude wanted
Yes
Have we printed phase yet
No
Page 2
Yes
No
Have we printed all outputs
Yes
Return
No
Page 2

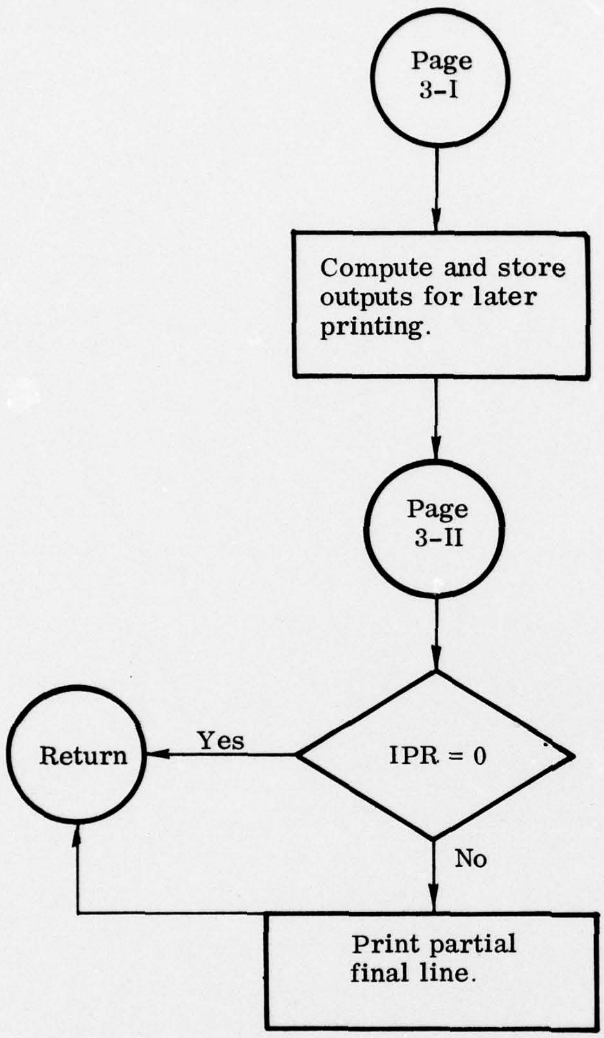
SENSPR-1

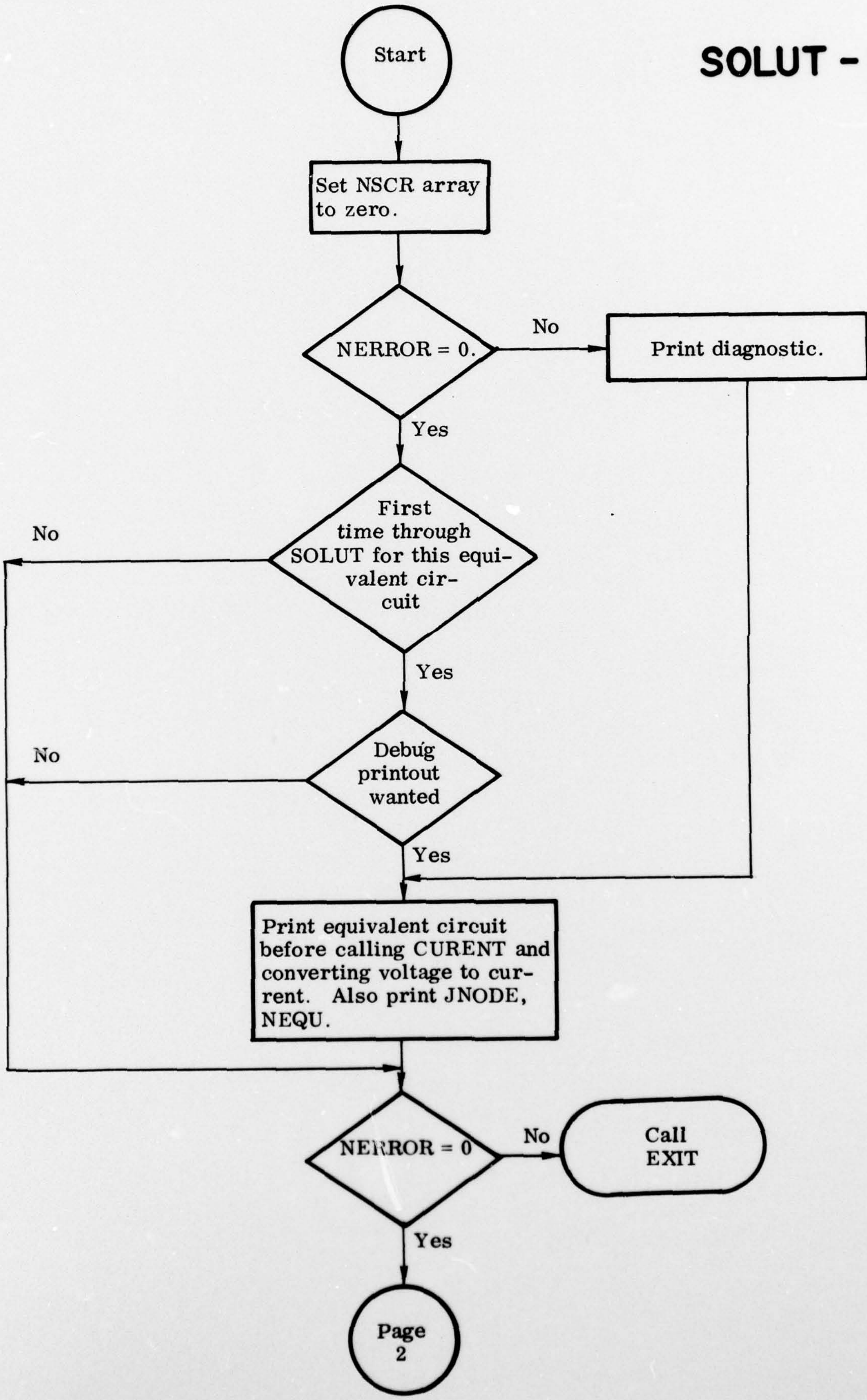

Start
IPR = 0
Examine NOUT array for non-zero entries in equivalent circuit.
IPR = IPR + 1
IPR greater than 8
Yes
Print a line of answers (actually four lines).
No
IPR = 1
Compute outputs and store for later printing.
Examine each C. E. in NCUR array to see if we should print it.
IPR = IPR + 1
Page 2

SENSPR- 2

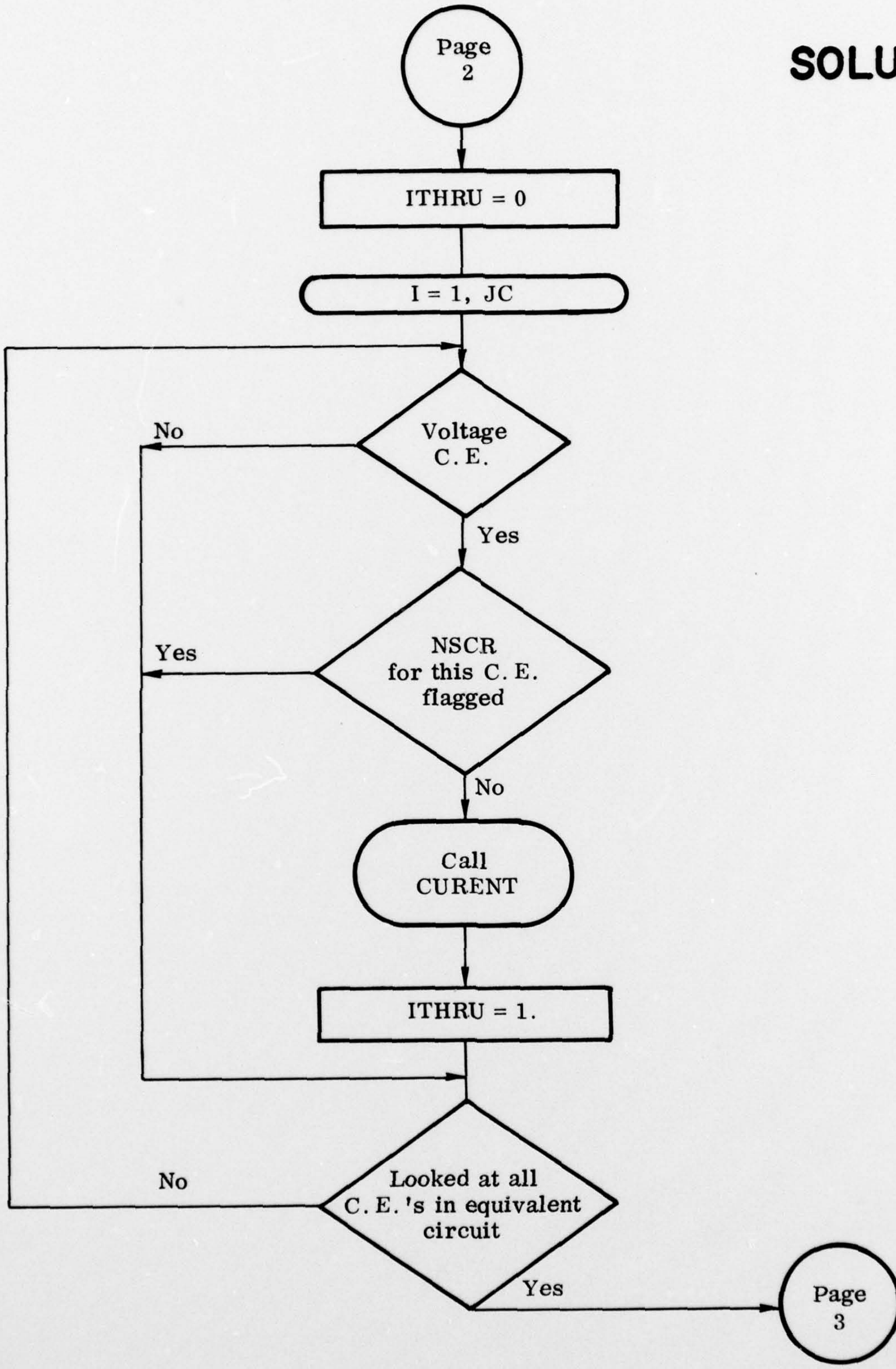


Page 2
IPR >8
Yes
Print four lines of eight outputs.
No
IPR = 1
Store output hi, low, and (Hi-Low) ÷ NOM plus titles in array for later printing.
Any special equation outputs
No
Page 3-II
Yes
Compute them.
IPR = IPR + 1
IPR >8
Yes
Print three lines of eight outputs.
No
IPR = 1
Page 3-I

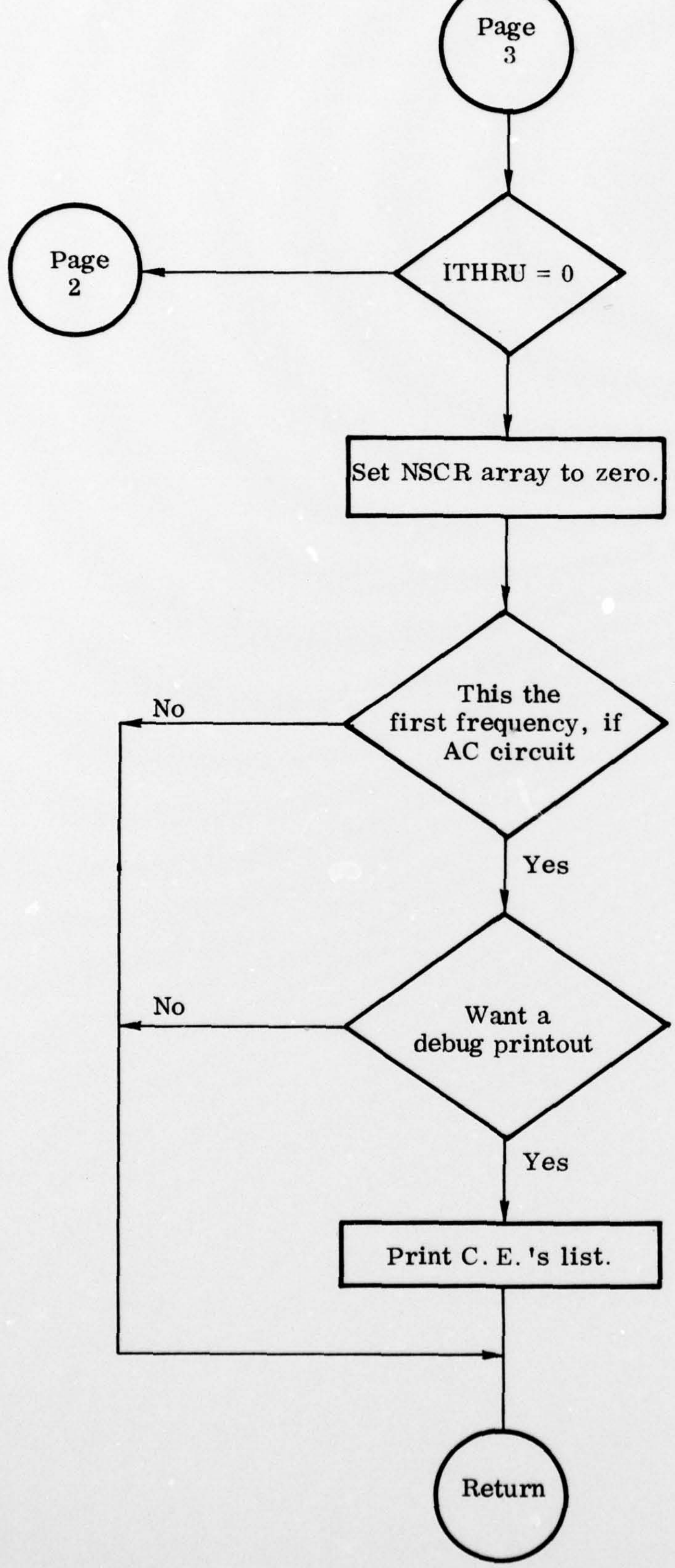

Page
3-I
Compute and store outputs for later printing.
Page
3-II
IPR = 0
Yes
Return
No
Print partial final line.


Start
Set NSCR array to zero.
NERROR = 0.
No
Print diagnostic.
Yes
First time through SOLUT for this equivalent circuit
No
Yes
Debug printout wanted
No
Yes
Print equivalent circuit before calling CURENT and converting voltage to current. Also print JNODE, NEQU.
NERROR = 0
No
Call EXIT
Yes
Page 2

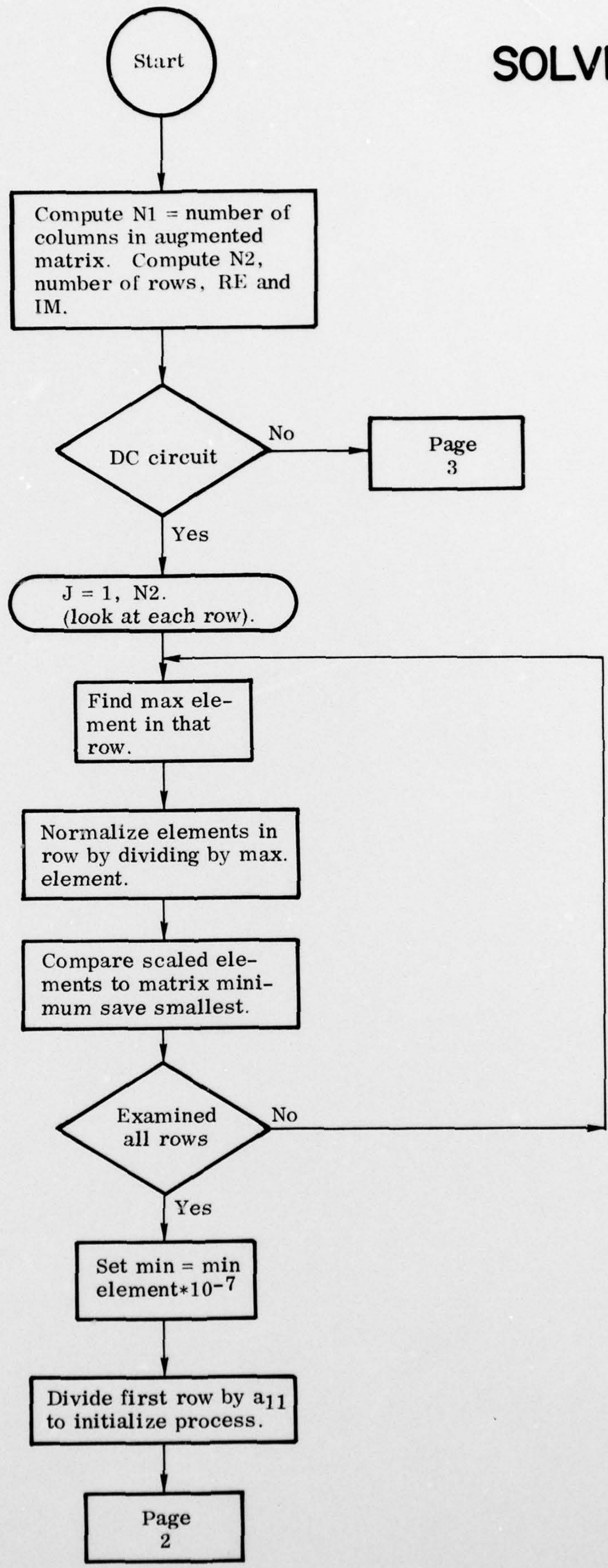
SOLUT - 2



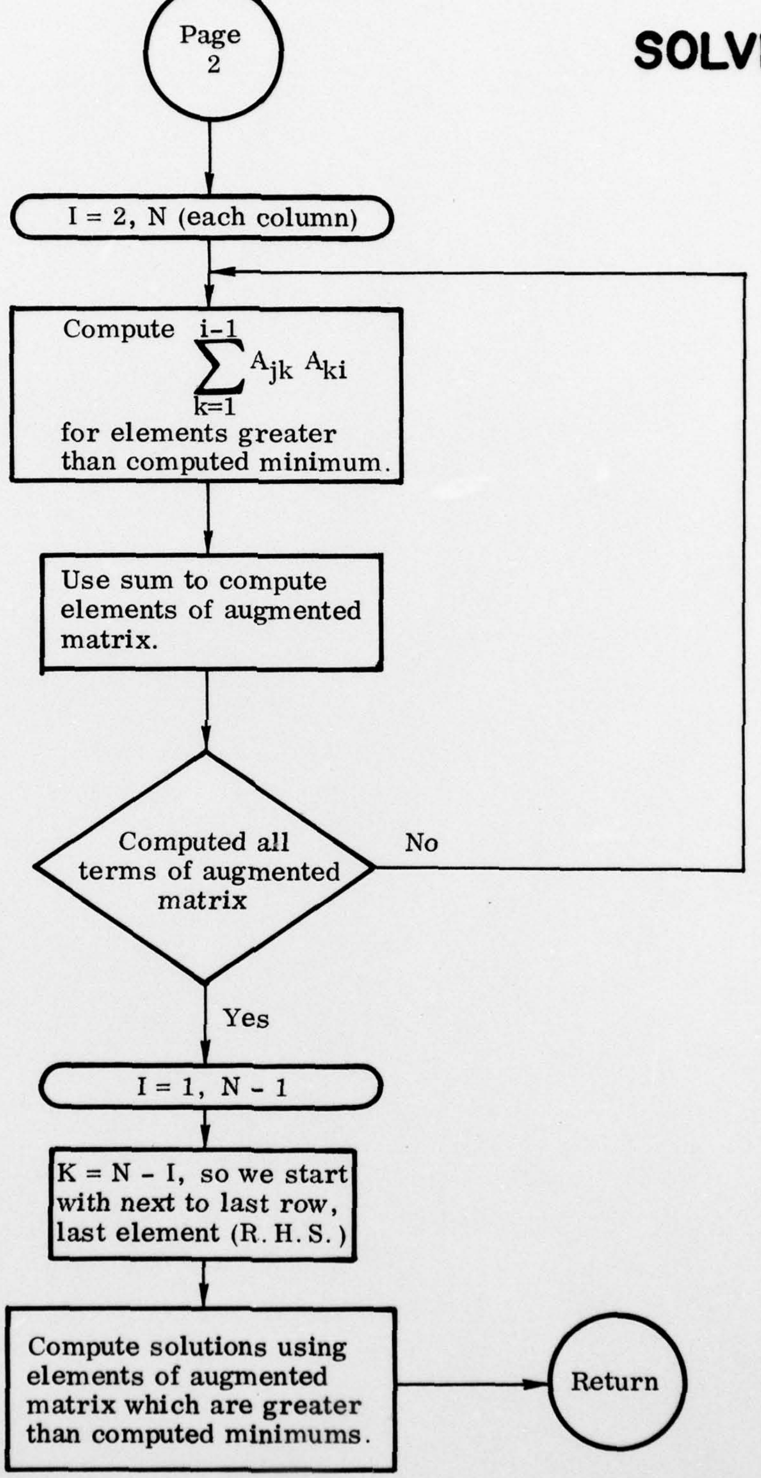
Page 2
ITHRU = 0
I = 1, JC
Voltage C.E.
No
Yes
NSCR for this C.E. flagged
Yes
No
Call CURENT
ITHRU = 1.
Looked at all C.E.'s in equivalent circuit
No
Yes
Page 3


Page 3
Page 2
ITHRU = 0
Set NSCR array to zero.
This the first frequency, if AC circuit
No
Yes
Want a debug printout
No
Yes
Print C. E. 's list.
Return

SOLVE-1

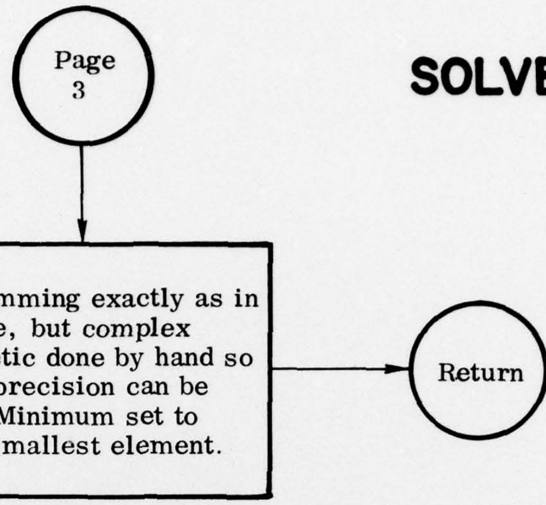

Start
Compute N1 = number of columns in augmented matrix. Compute N2, number of rows, RE and IM.
DC circuit
No
Page 3
Yes
J = 1, N2. (look at each row).
Find max element in that row.
Normalize elements in row by dividing by max. element.
Compare scaled elements to matrix minimum save smallest.
Examined all rows
No
Yes
Set min = min element*10-7
Divide first row by a11 to initialize process.
Page 2

SOLVE - 2

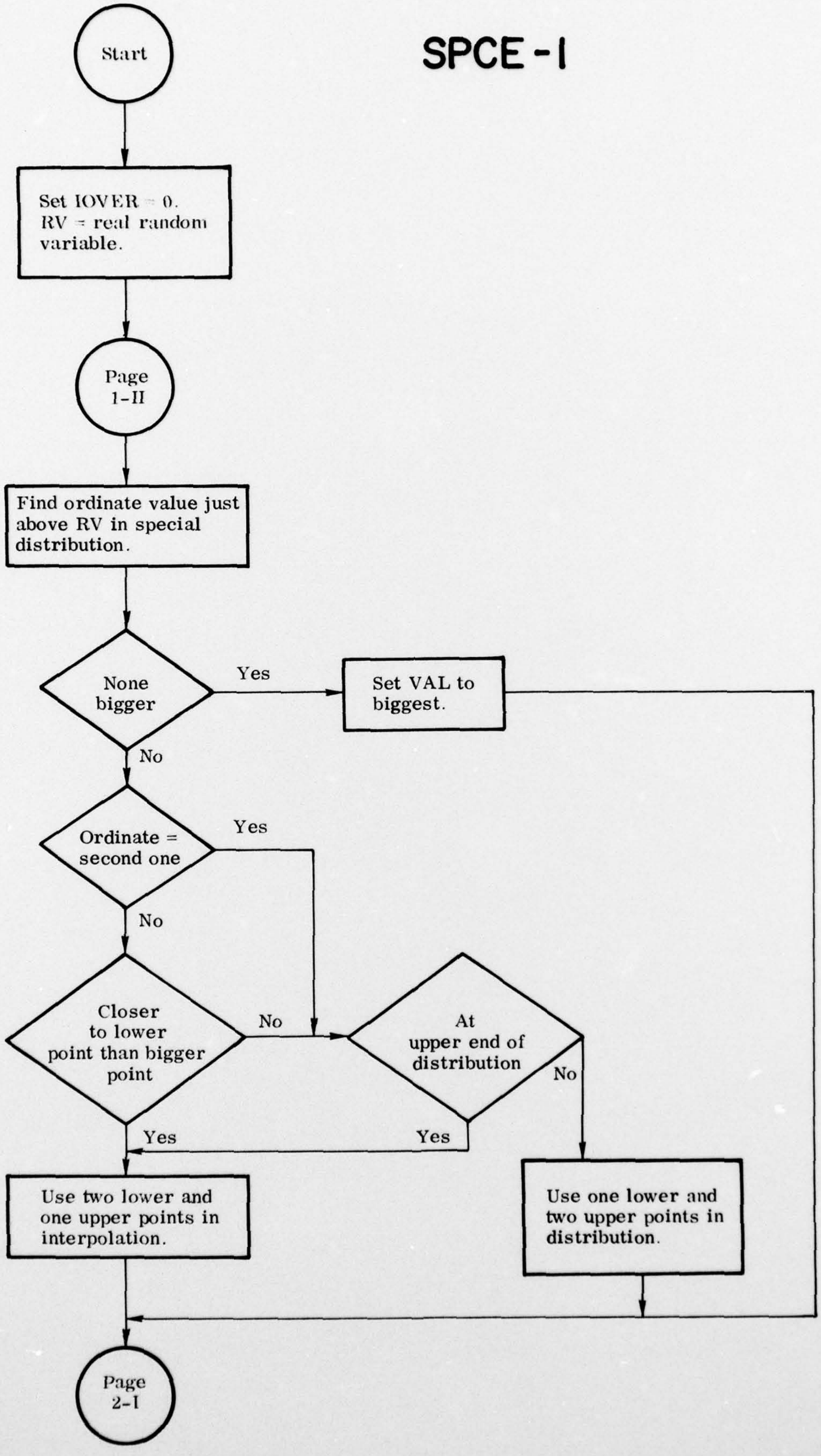


Page 2
I = 2, N (each column)
Compute $\sum_{k=1}^{i-1} A_{jk} A_{ki}$ for elements greater than computed minimum.
Use sum to compute elements of augmented matrix.
Computed all terms of augmented matrix
No
Yes
I = 1, N - 1
K = N - I, so we start with next to last row, last element (R. H. S.)
Compute solutions using elements of augmented matrix which are greater than computed minimums.
Return

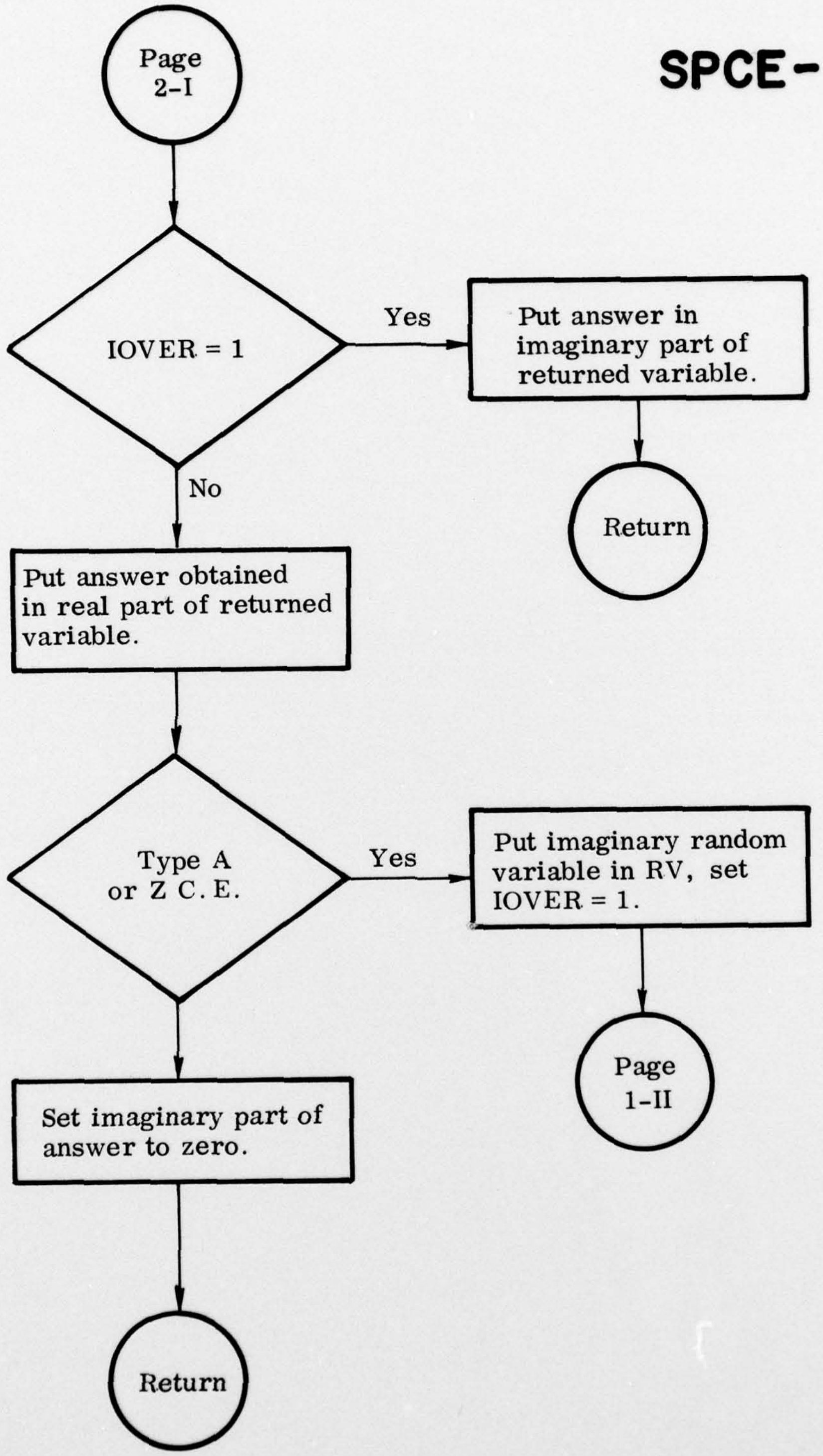
SOLVE - 3


Page 3
Programming exactly as in DC case, but complex arithmetic done by hand so double precision can be used. Minimum set to 10^{-17}*smallest element.
Return

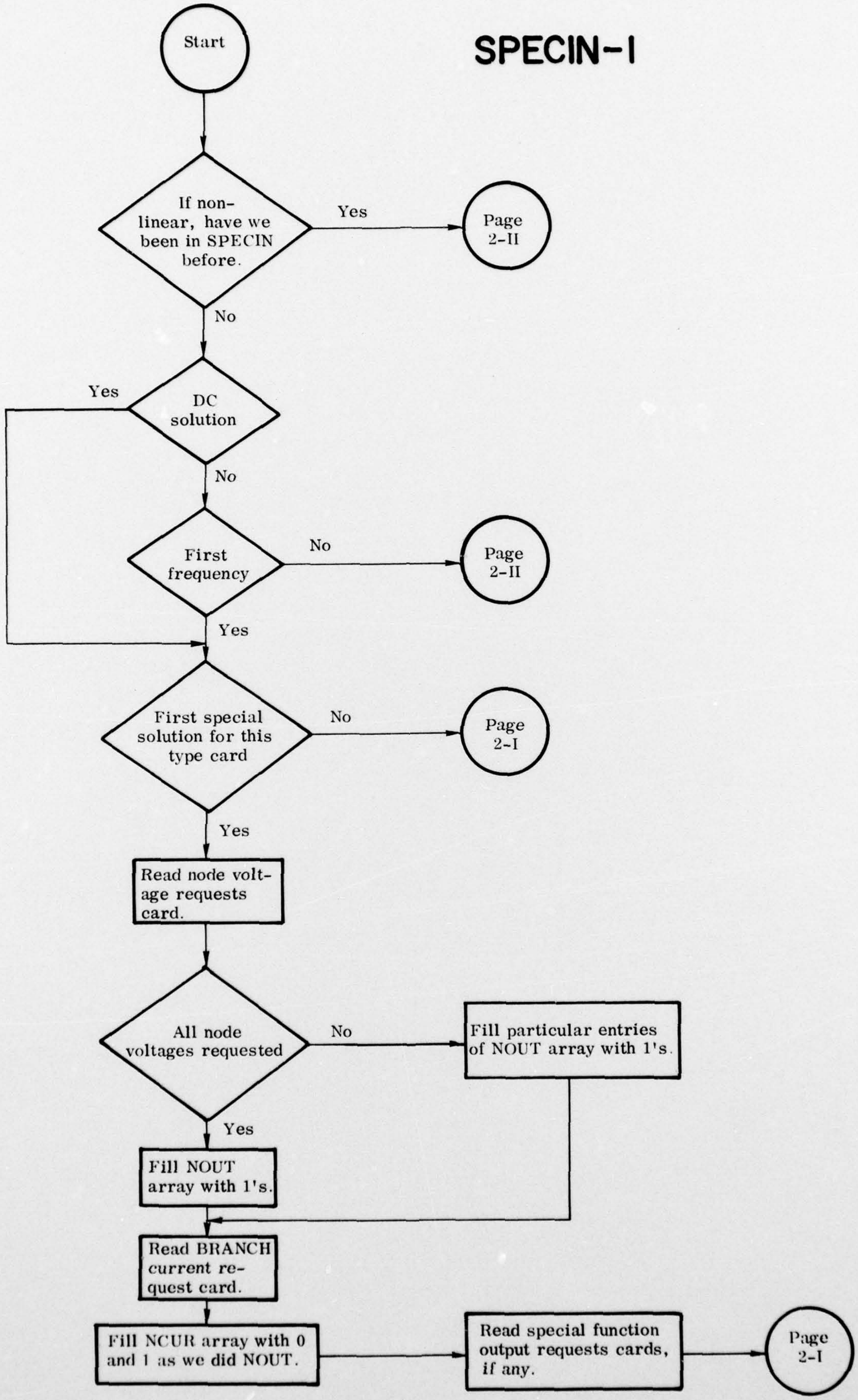
SPCE-1

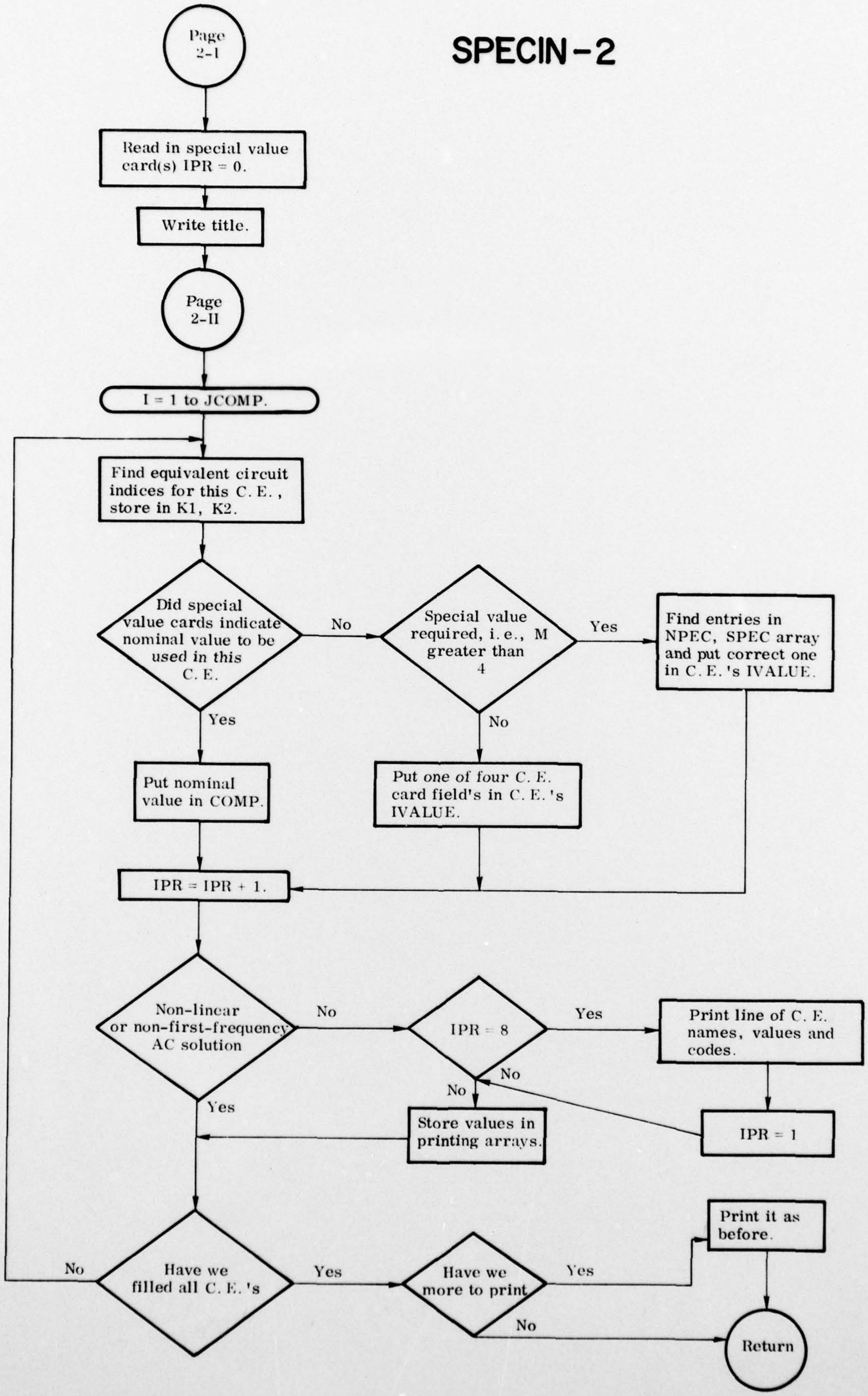

Start
Set IOVER = 0.
RV = real random variable.
Page 1-II
Find ordinate value just above RV in special distribution.
None bigger
Yes
Set VAL to biggest.
No
Ordinate = second one
Yes
No
Closer to lower point than bigger point
No
At upper end of distribution
No
Yes
Yes
Use two lower and one upper points in interpolation.
Use one lower and two upper points in distribution.
Page 2-I

SPCE-2

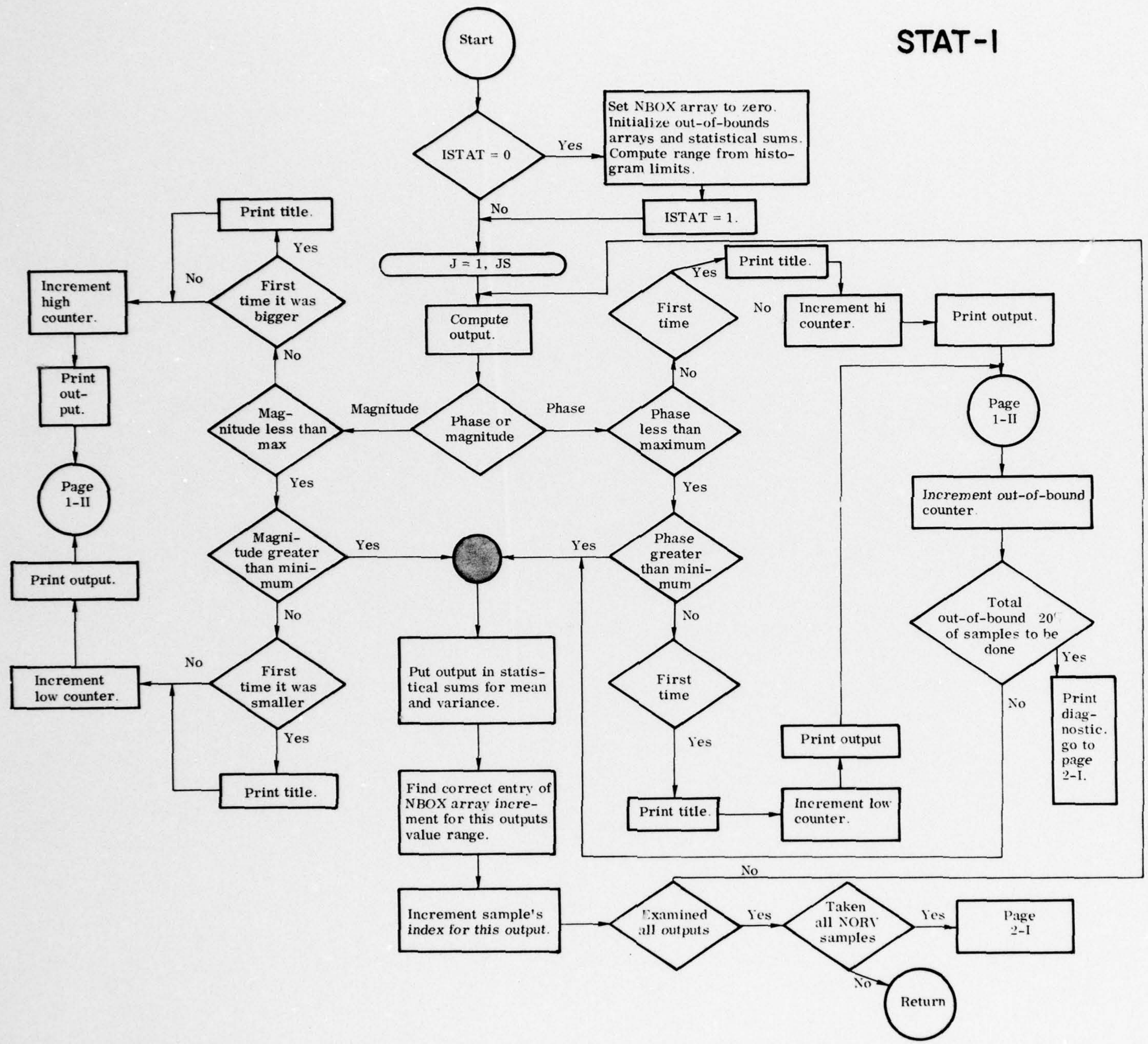

Page
2-I
IOVER = 1
Yes
Put answer in imaginary part of returned variable.
Return
No
Put answer obtained in real part of returned variable.
Type A or Z C. E.
Yes
Put imaginary random variable in RV, set IOVER = 1.
Page
1-II
Set imaginary part of answer to zero.
Return

SPECIN-I

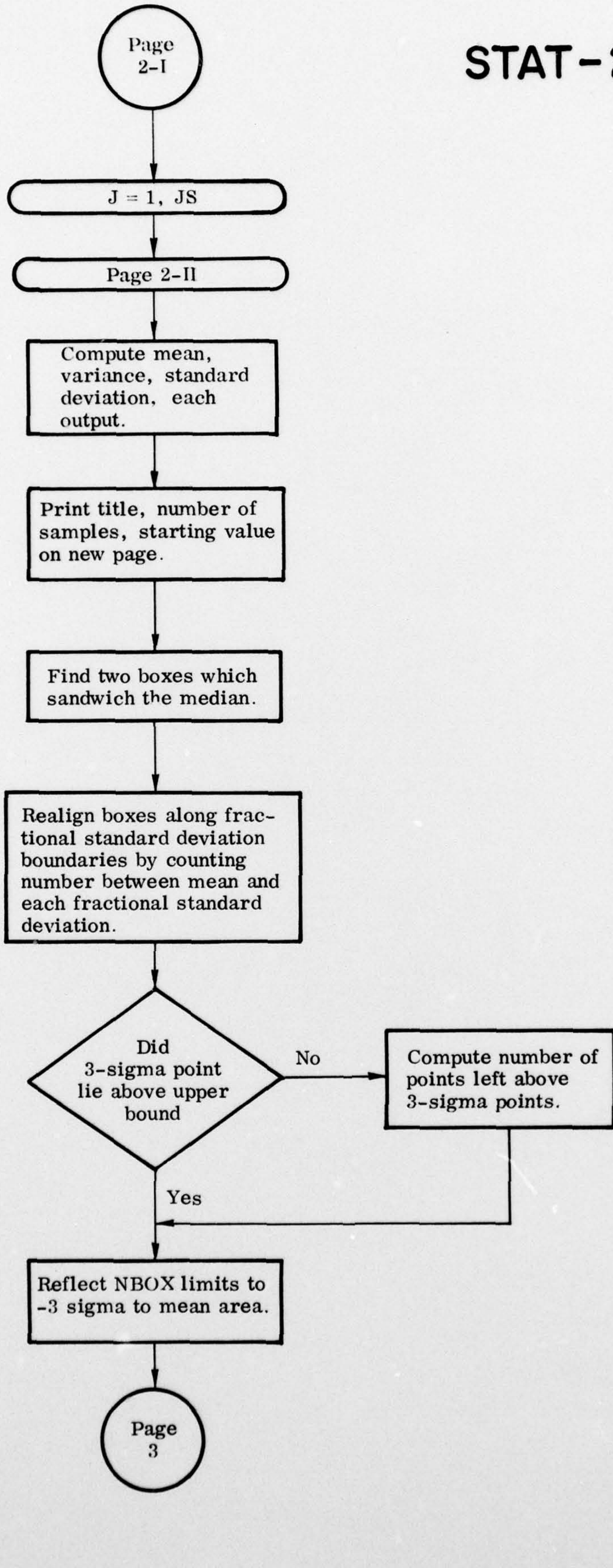

Start
If non-linear, have we been in SPECIN before.
Yes
Page 2-II
No
DC solution
Yes
No
First frequency
No
Page 2-II
Yes
First special solution for this type card
No
Page 2-I
Yes
Read node voltage requests card.
All node voltages requested
No
Fill particular entries of NOUT array with 1's.
Yes
Fill NOUT array with 1's.
Read BRANCH current request card.
Fill NCUR array with 0 and 1 as we did NOUT.
Read special function output requests cards, if any.
Page 2-I


SPECIN-2
Page 2-I
Read in special value card(s) IPR = 0.
Write title.
Page 2-II
I = 1 to JCOMP.
Find equivalent circuit indices for this C. E., store in K1, K2.
Did special value cards indicate nominal value to be used in this C. E.
No
Special value required, i. e., M greater than 4
Yes
Find entries in NPEC, SPEC array and put correct one in C. E. 's IVALUE.
Yes
Put nominal value in COMP.
No
Put one of four C. E. card field's in C. E. 's IVALUE.
IPR = IPR + 1.
Non-linear or non-first-frequency AC solution
No
IPR = 8
Yes
Print line of C. E. names, values and codes.
No
No
Yes
Store values in printing arrays.
IPR = 1
Print it as before.
No
Have we filled all C. E. 's
Yes
Have we more to print
Yes
No
Return

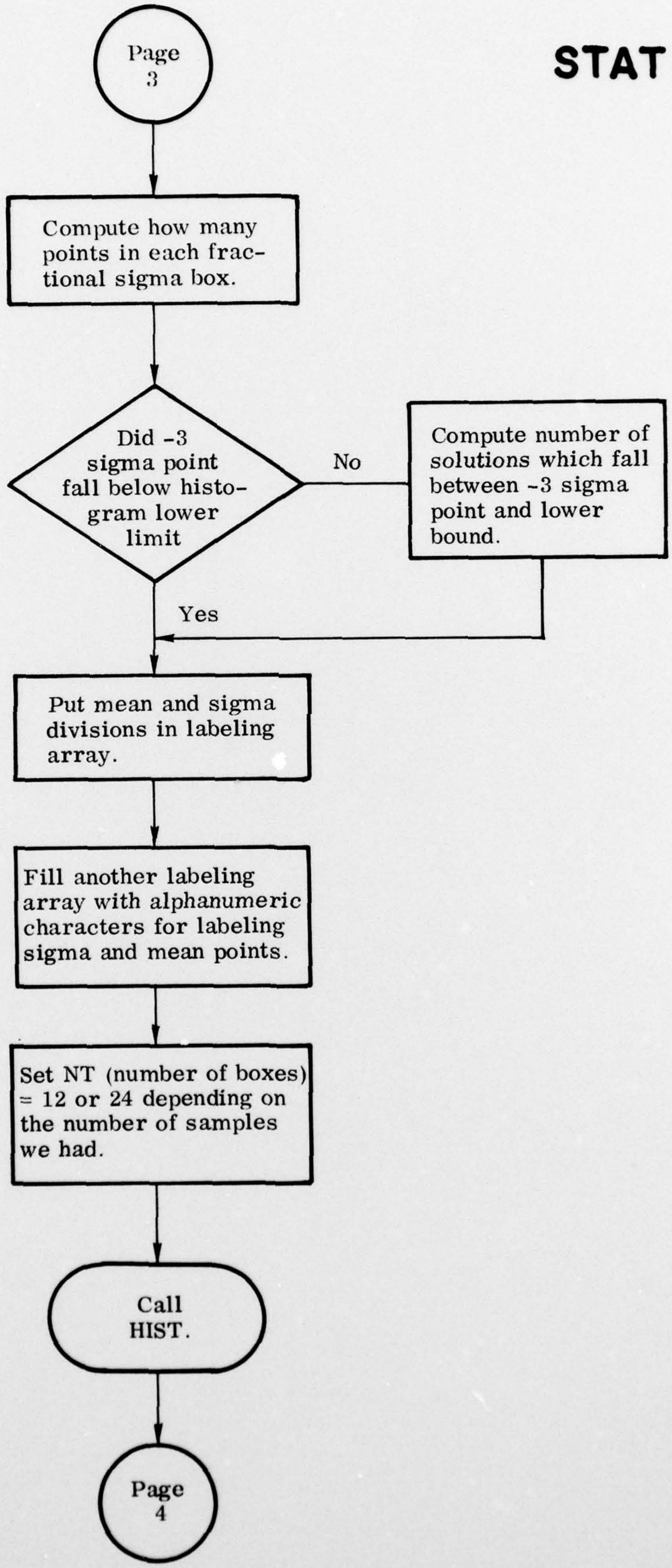
STAT-I


Start
ISTAT = 0
Yes
Set NBOX array to zero. Initialize out-of-bounds arrays and statistical sums. Compute range from histogram limits.
ISTAT = 1.
No
J = 1, JS
Compute output.
Phase or magnitude
Magnitude
Phase
Mag-nitude less than max
No
First time it was bigger
Yes
Print title.
No
Increment high counter.
Print out-put.
Page 1-II
Yes
Magni-tude greater than mini-mum
Yes
No
First time it was smaller
No
Yes
Print title.
Increment low counter.
Print output.
Page 1-II
Phase less than maximum
No
First time
Yes
Print title.
No
Increment hi counter.
Print output.
Page 1-II
Increment out-of-bound counter.
Total out-of-bound 20% of samples to be done
Yes
Print diag-nostic. go to page 2-I.
No
Yes
Phase greater than mini-mum
Yes
No
First time
Yes
Print title.
Increment low counter.
Print output
Put output in statistical sums for mean and variance.
Find correct entry of NBOX array increment for this outputs value range.
Increment sample's index for this output.
Examined all outputs
No
Yes
Taken all NORV samples
Yes
Page 2-I
No
Return

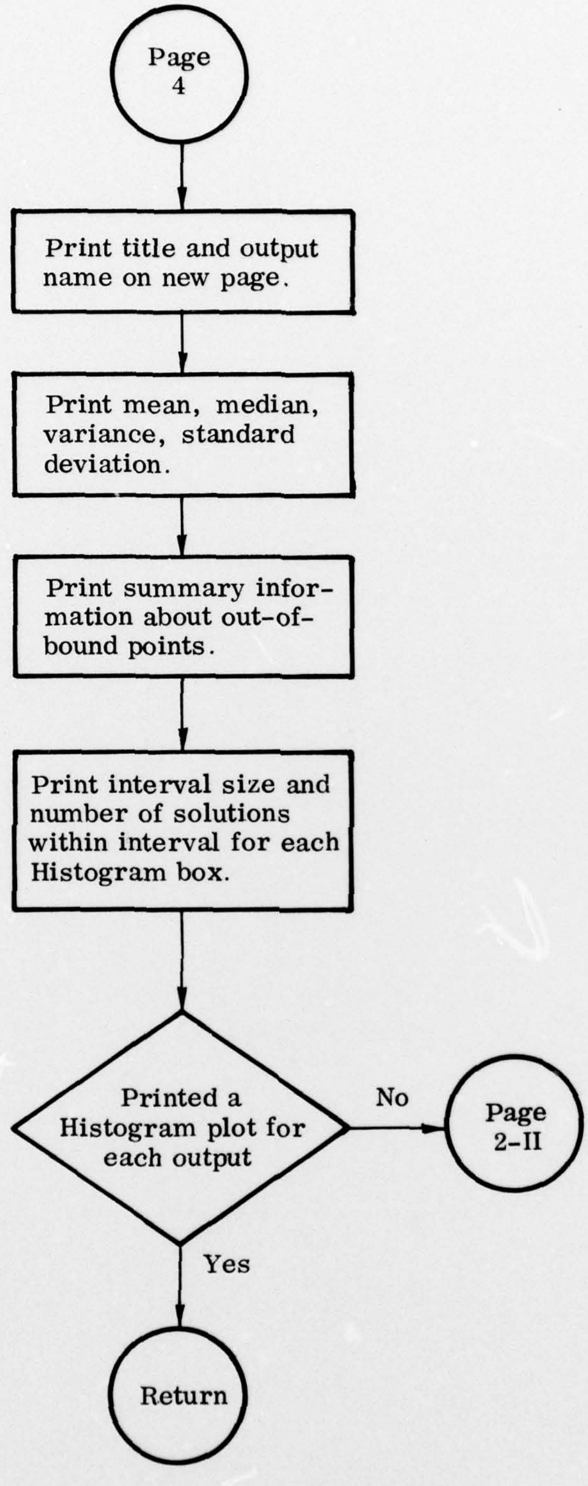
STAT-2


Page 2-I
J = 1, JS
Page 2-II
Compute mean, variance, standard deviation, each output.
Print title, number of samples, starting value on new page.
Find two boxes which sandwich the median.
Realign boxes along fractional standard deviation boundaries by counting number between mean and each fractional standard deviation.
Did 3-sigma point lie above upper bound
No
Compute number of points left above 3-sigma points.
Yes
Reflect NBOX limits to -3 sigma to mean area.
Page 3

STAT - 3


Page 3
Compute how many points in each fractional sigma box.
Did -3 sigma point fall below histogram lower limit
No
Compute number of solutions which fall between -3 sigma point and lower bound.
Yes
Put mean and sigma divisions in labeling array.
Fill another labeling array with alphanumeric characters for labeling sigma and mean points.
Set NT (number of boxes) = 12 or 24 depending on the number of samples we had.
Call HIST.
Page 4

STAT-4


Page 4
Print title and output name on new page.
Print mean, median, variance, standard deviation.
Print summary information about out-of-bound points.
Print interval size and number of solutions within interval for each Histogram box.
Printed a Histogram plot for each output
No
Page 2-II
Yes
Return